Antti Pohjalainen

# Control Policies of an Automated Storage and Retrieval System

**School of Electrical Engineering**

Thesis submitted for examination for the degree of Master of Science in Technology
Espoo 25.9.2015

**Instructor:** M.Sc. Martti Peuransalo

**Supervisor:** Docent Kai Zenger

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Antti Pohjalainen

Title: Control Policies of an Automated Storage and Retrieval System

Date: 25.9.2015 Language: English Number of pages: 7 + 61

Department of Electrical Engineering and Automation

Professorship: Control Engineering Code: AS-74

Thesis supervisor**:** Docent Kai Zenger

Thesis advisor: M.Sc.(Tech.) Martti Peuransalo

Automated storage and retrieval systems (AS/RS) are popular for storing semi-fast moving items in distribution centers. They are costly systems whose design involves many critical decisions, which affect the overall performance of the system.

This work is focused on the crane control policies of a double-deep dual-shuttle AS/RS. The goal of the thesis is to find out how operating performance of a crane can be improved with storage location assignment, dwell-point positioning and request sequencing.

Alternative control rules were developed based on AS/RS literature and tested against those implemented by the supplier of the system. The comparison was carried out with a discrete-event simulation tool, which was built as a part of the thesis. The system was simulated under two different workload scenarios and rack fill levels.

The simulation results indicate that sequencing and cycle formation algorithms can have a significant effect on system throughput in periods of high utilization. The effect was found larger with the lower 70 % fill level. The linear programming sequencing algorithm developed in this thesis was found to reduce the average cycle time by 5.3 % compared to the algorithm used by the supplier.

In the on-shift scenario, the optimal dwell point strategy could reduce the average crane response time by 10 % compared to the policy used by the supplier. However, this difference was not noticeable when the average request turnover time was used as a measure.

**Keywords:** automated storage and retrieval system, crane control, discrete-event simulation

AALTO-YLIOPISTO
SÄHKÖTEKNIIKAN KORKEAKOULU

DIPLOMITYÖN
TIIVISTELMÄ

Tekijä: Antti Pohjalainen

Työn nimi: Automaattisen varastointi- ja keruujärjestelmän ohjausperiaatteet

Päivämäärä: 25.9.2015 Kieli: Englanti Sivumäärä: 7 + 61

Sähkötekniikan ja automaation laitos

Professuuri: Systeemitekniikka Koodi: AS-74

Työn valvoja**:** Dosentti Kai Zenger

Työn ohjaaja: Dipl. ins. Martti Peuransalo

Jakelukeskuksissa yleiset hyllystöhissijärjestelmät ovat kalliita investointeja. Niiden keräilytehoa voidaan osaltaan parantaa tehostamalla varastohissin ohjausmenetelmiä.

Tämän diplomityön tutkimuksen kohteena on hyllystöhissijärjestelmä, jossa on tuplasyvät hyllyt sekä hissi, jolla on kahden laatikon kantokapasiteetti. Työn tavoitteena oli selvittää, miten järjestelmän suorituskykyä voidaan parantaa hyllypaikkojen allokoinnin, hissin odotuspaikan valinnan, sekä hissitehtävien sekvensoinnin avulla.

Järjestelmätoimittajan ohjausperiaatteiden hyvyyttä arvioitiin vertaamalla niitä kirjallisuuden pohjalta kehitettyihin ohjausmenetelmiin. Ohjausten vertailu tehtiin simulointityökalulla, joka rakennettiin työn aikana. Testiskenaarioissa simuloitiin järjestelmää kahdessa eri kuormitustilanteessa ja kahdella eri täyttöasteella.

Simulointitulosten perusteella sekvensointi- ja syklinmuodostusalgoritmeillä huomattiin olevan merkittävä vaikutus tilanteissa, joissa hissillä on tehtäväjonoja. Vaikutus oli suurempi matalammalla 70 % täyttöasteella. Työssä kehitetty sekalukuoptimointiin perustuva sekvensointialgoritmi lyhensi keskimääräistä sykliaikaa 5,3 % toimittajan käyttämään algoritmiin verrattuna.

Matalan käyttöasteen skenaariossa optimaalisen odotuspaikan valinta lyhensi hissin liikkumisaikaa seuraavan tehtävään 10 %:llä, mutta ero oli käytännössä olematon, kun mittarina käytettiin keskimääräistä aikaa tehtävän saapumisesta sen suorittamiseen.

**Avainsanat:** automaattivarasto, tapahtumapohjainen simulointi, ohjausalgoritmi, optimointi

# Preface

This thesis was conducted at EP-Logistics Ltd. It was inspired by a significant ongoing warehouse automation project, in which I am lucky to have been a part of since an early stage. I am very grateful for having received the opportunity to conduct an interesting and somewhat independent study with connection to this exciting project.

First of all, I would like to thank my instructor Martti Peuransalo for helping me sort out my thoughts on many occasions, and giving me numerous ideas and pieces of advice throughout the many phases of this work. Martti's helpfulness and profound interest in this thesis was both motivating and encouraging.

I want to thank Pekka Korpiharju for sharing his expertise on the topic and giving very comprehensive and helpful feedback on the work. Pekka also pointed me to interview Mr. Folke Wahlström, to whom I am grateful for sharing his broad knowledge of warehouse automation systems and their control implementations.

I would also like to thank my supervisor Kai Zenger for guiding me through the process of writing the thesis and responding quickly to my questions. Finally, I appreciate the all the support that I have received from my family and friends prior to, and during this thesis.

Helsinki, 20.9.2015

Antti Pohjalainen

# Contents

# Symbols and Abbreviations

## Symbols

$b$ shape factor

$f$ frozen horizon

$h$ sequencing horizon

$p_s$ probability of a storing task after idle period

$s$ fit parameter of the ABC curve

## Abbreviations

AS/RS automated storage and retrieval system

COI cube-per-order index

COL closest-open-location

FCFS first-come-first-serve

FIFO first-in-first-out

KPI key performance indicator

LP linear programming

LTPR locations-to-product ratio

MILP mixed integer linear program

SKU stock keeping unit

WCS warehouse control system

WMS warehouse management system

# 1 Introduction

## 1.1 Background

Automated storage and retrieval systems (AS/RS) have become an increasingly important part of material flow handling in distribution centers and flexible manufacturing systems. The benefits of AS/R systems include high storage density, fast and reliable storing and retrieving with minimal human intervention, and real time inventory tracking. These characteristics lead to direct advantages over non-automated systems, including savings in labor costs and floor space and reduced error rates. [1] The main drawbacks of AS/R systems include high investment costs, inflexible layout, and limited capacity. [1][2]

The early applications of AS/RSs mostly involved *unit-load* cranes, which were used to handle heavy pallets of finished goods, weighing 1000-3000 kg [3, p.648]. The design of these systems was mainly concerned with storage space, only little attention was given to the effectiveness of system operation. Since then, developments such as the strong development of e-commerce, have brought more focus on the design and management of order picking systems. Short and precise delivery times according to customer needs necessitate higher throughput and faster response times. This is particularly important in E-fulfillment warehouses where thousands of business-to-consumer orders with small quantities have to be processed daily. [4]

The technology of AS/R systems has also advanced a lot in the past few decades. Current designs allow crane-based *miniload* machines to operate reliably at speeds up to 350 m/min with accelerations and decelerations up to 0.8 g [3, p.644]. Miniload systems are used for storing and retrieving small items, e.g. consumer electronics or food. The items are stored in totes, which can be subdivided into compartments, each containing one product. Due to their high operating speeds and handling capacity, miniloads are well suited for automated distribution and manufacturing processes.

The high cost of AS/R systems makes it important to put effort into maximizing system productivity in order to increase the return of the investment. When an AS/R system is designed, one has to address many issues related to physical design and control. Both can have a significant effect on operating performance. This work is concerned with the crane control policies of the system. Control in this case means the logic, which governs the movements of a storage and retrieval machine. Although AS/R systems have been seemingly thoroughly studied over the past few decades, a majority of these studies has focused on the basic system type, the unit-load AS/RS [1][5].

Discrete-event simulation has been used in AS/R study for at least three decades [6]. It has proven widely useful in verifying analytical models, as well as studying the effects of control policies with various AS/RS configurations. The first main benefit of simulation is that it makes comparison of different AS/RS configurations effective. With simulation it is also possible to separate the effects of physical design and control issues [7]. Finally, simulation provides a good means to analyze how the system operates under different

rack fill levels and stochastic demand patterns. In a warehouse automation project, simulation is often applied in the early stages of the planning phase, when comparing different layout and material handling equipment solutions. These models are frequently implemented in little time, from nearly scratch. This leads to an abundance of assumptions and modeling simplifications. The simplifications are often justified when the scope of the models is wide and the results are used for a high level analysis of e.g. a complete warehousing system. In this thesis, simulation is used for a detailed analysis of controls of one subsystem of an automatic storage system.

This thesis was conducted at EP-Logistics Ltd. EP-Logistics is a logistics consulting company that has been involved in many warehouse automation projects from the early planning phase throughout the implementation. The system which is studied in the thesis is part of an ongoing automated third-party logistics warehouse project. The automated storage system is delivered by a major logistics systems supplier. The AS/RS subsystem related to this project will be referred to as the installed system in subsequent chapters.

## 1.2 Goals and research questions

The main goal of the thesis is to develop and test alternative methods to control the crane movements of a double-deep, dual-shuttle AS/R system, and to compare the efficiency of these methods with those applied in a system which is being implemented in an ongoing storage automation project. Another goal is to build a simulation tool, which can be used for analyzing the performance of different AS/RS configurations, including the installed system type, with a varying set of crane control rules. To that end, both physical design issues as well as control methods are parametrized in the tool. In this work, the implemented simulation tool is only used for analysis of the installed system.

The control decisions which are considered in the work are storage location assignment, dwell point selection, and request sequencing. Most consideration will be given to request sequencing and cycle formation algorithms because their effect can be studied without information or assumptions of the SKUs (store keeping unit) that will be stored in the system. The research questions that will be answered in the thesis are:

- What alternative crane control methods could be used to improve those implemented by the supplier?
- How much does the choice of crane controls influence the installed system under different rack fill levels and system utilization rates?

The latter question will be answered by modeling the installed AS/RS and its control decisions by means of discrete-event simulation. The simulation model will provide a means to quantify the effects of control decisions. The former question will be answered by adapting and applying control methods found in AS/RS literature and comparing their performance with the controls used by the supplier. The simulation model utilized for the comparison.

The results of this thesis give insight to how effectively the crane control implemented by the supplier functions and how it could be improved. The simulation tool implemented in the thesis can be used at EP-Logistics in future projects for analyzing other types of AS/RS configurations with little or no modification.

## 1.3 Structure of the work

The rest of this thesis is structured as follows: Chapter 2 will give a brief system description of AS/RSs, their operating principles, performance measurement and physical design issues. Chapter 3 will present the crane control policies used for AS/RS control. Solution algorithms for the sequencing and cycle formation problem will be formulated in chapter 4.

Chapter 5 will present the discrete-event simulation tool, which was built for the purpose of evaluating AS/RS design and control decisions. The test scenarios are then formed, and the results of the simulation runs presented and analyzed in chapter 6. Finally, the thesis is concluded in chapter 7.

# 2 Automated storage and retrieval systems

This chapter will introduce the basic terminology and operating principles of an AS/RS. Performance evaluation will be discussed in final section of the chapter.

## 2.1 System description and configurations

**Description**

The function of an AS/RS is to automatically transfer items between high storage racks and picking or processing stations. A system, as presented in Figure 1, consists of three main components: *storage racks*, *cranes* (S/R machine) and *I/O points*. Cranes are fully automated storage and retrieval machines that can autonomously move, pick up and drop off totes. They operate in aisles between two storage racks. In most configurations the cranes operate in dedicated aisles. The movements of a crane are controlled with an industrial PC or PLC (programmable logic controller). Each crane has at least three independent frequency-controlled AC drives: one for horizontal movement, one for vertical movement, and one for operating the load handling device also known as a shuttle [8]. The shuttles can only move complete loads, usually totes or pallets, instead of handling single pieces of an item.

The storage racks are stationary and rectangular. They are located on either side of a crane. The storage locations are equally sized. There is an I/O area at the end of each aisle, which serves as both the retrieval point for storing tasks and drop off point for retrieval tasks. The I/O point is usually located near the lower corner of the rack [9].

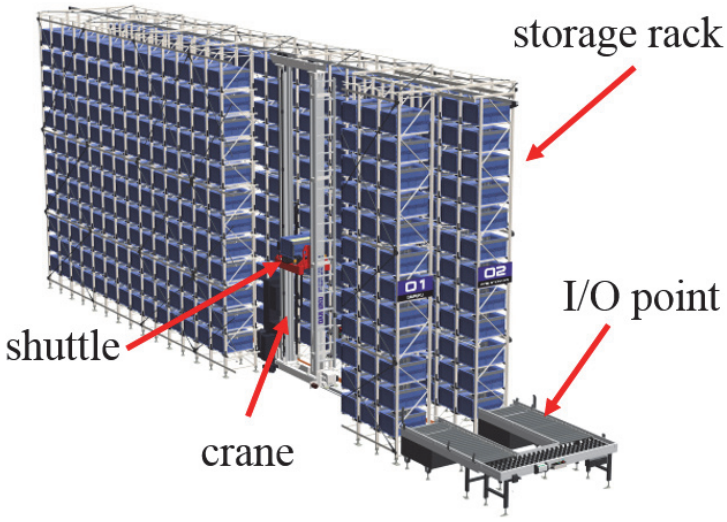


Figure 1: An automated storage and retrieval system [10]

Figure 2 displays a typical parts-to-picker system, commonly found in distribution centers. It consists of an $n$-aisle AS/RS interfaced to $m$ picking stations with a conveyor loop. The loop acts as a buffer and decouples the AS/RS from the picking stations. The installed system also has this basic structure. The loop is also connected to an infeed

station for replenishing the storage, and an outfeed for empty storage totes. In the installed system, there is also a second conveyor loop for order totes, which conveys the picked goods to a packing and sorting area.

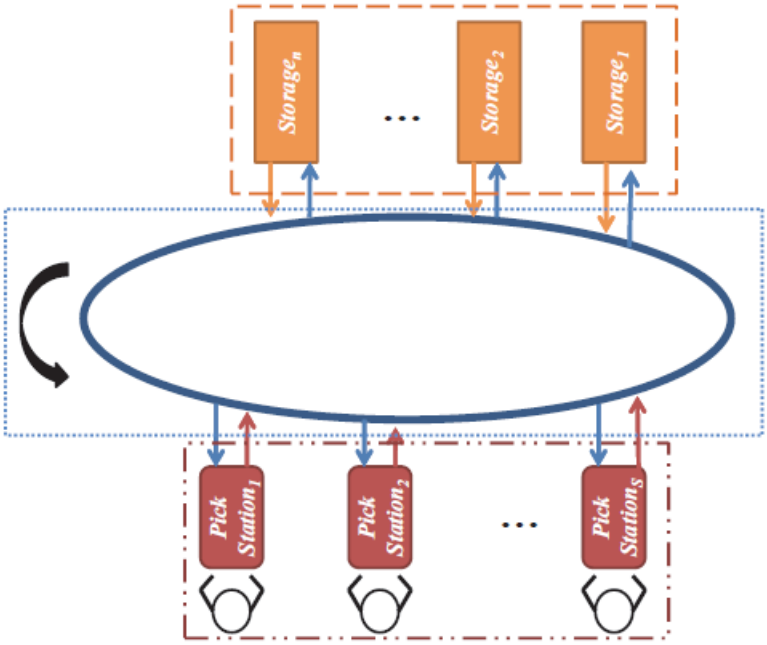


Figure 2: A parts-to-picker system [11]

**Configurations**

Various AS/RS configurations have been developed for different needs. A representation of the most important configuration parameters and their common values is presented in Table 1 [9]. These parameters define the system type, which strongly affects the choice of control policies.

Table 1: Typical system configuration parameters of AS/R systems. [9] The values related to the installed system are bolded.

| Configuration parameter | Specifications | | |
|---|---|---|---|
| Depth of rack | Single-deep | | **Double-deep** |
| Number of shuttles on a crane | 1 | **2** | 3 |
| Crane operating range | **Dedicated-aisle** | | Access to multiple aisles |
| Position of the I/O point | In the corner of the rack | | **Offset in x- or y-direction** |

The installed system has double-deep storage racks. They are commonly used to increase the storage capacity of an AS/RS [12]. In a double-deep rack, each storage location has the capacity to store two totes, one in front of the other. In this thesis, the tote places will be called the front and back positions. Double-deep storage provides a 50 % saving in aisle space compared to single-deep racks. On the other hand, retrieval costs are generally higher with double-deep storage due to forced rearrangements [9]. A forced rearrangement has to be made whenever a tote is retrieved from the back position and there is another tote, which is not currently needed, stored in front of it. The blocking tote needs to be moved to another empty location. [12] This rearranging naturally causes extra travel for the crane. A higher fill level means less empty spaces and more forced rearrangements. Because some locations have to be kept empty for rearrangements, the rack fill level should never be too high. The supplier has set an upper limit of 90 % for the fill level. Under normal operation, the fill level should be under this limit, but over 50 % to make use of the storage capacity provided by the extra locations.

Another method to increase the storage capacity in terms of different SKUs is to use compartmented storage totes. This means that multiple SKUs can be stored in one storage tote. In the installed system the SKUs are stored in totes with one, two, four or eight compartments. The percentage of different types of totes in the system is shown in Table 2.

Table 2: Percentage of different storage totes in the system

| number of compartments | % of totes |
|---|---|
| 1 | 78.7 |
| 2 | 9.8 |
| 4 | 4.7 |
| 8 | 6.7 |

The number of shuttles on a crane is also an important configuration parameter. *Single-shuttle cranes* can handle one tote at a time. To increase handling capacity, *multi-shuttle cranes* have been developed. In general, system throughput increases when more shuttles are added on a crane, since the amount of empty travel decreases. However, there is a diminishing return on throughput for each additional shuttle because load handling time also increases [13]. Additional shuttles also bring on more costs. In practice, single- and dual-shuttle systems are common. Triple-shuttle systems rare, and practically no systems with more shuttles than three are used. [13] The installed system has dual-shuttle cranes. For a multi-shuttle system, it is also important to take into account, whether the shuttles can be operated independently. Independent operation means that each shuttle has its own AC drive. This enables the crane to perform a swap move, i.e. storing and retrieving sequentially from the same rack location. In the installed system the shuttles are operated with one shared drive, which means that performing a swap is not possible. This is an

important restriction because it means that all storing and retrieving tasks in the same cycle need to be executed in different rack locations.

The position of the I/O point can sometimes be located in another position than the corner of the rack. Some experiments have been made, where locating the I/O-point at the middle of the aisle has resulted in a higher throughput. It is also possible to have multiple I/O points in one aisle, as well as having a separate pickup and deposit location. [1] The installed system is a five-aisle AS/RS. There is one dedicated crane operating in each aisle. The I/O point is offset from the lower corner by four locations in the vertical axis, in location coordinates (0, 4). The configuration parameter values of the installed system are bolded in Table 1. In subsequent chapters, the unit-load system will also be mentioned several times, because it is the most researched and common system type. The unit-load AS/RS is essentially a single-shuttle, single-deep system.

## 2.2 Operating principles

Each crane has two sets of requests to serve: storing and retrieval tasks. The machine serves these requests by operating in *command cycles*. When there are multiple tasks in queue, a cycle starts and ends at the I/O point. After performing the last currently known request, the crane is driven to its *dwell point*, where it starts a new cycle upon the arrival of new requests. Command cycles can be classified by task type and number of totes handled. In a storing or retrieval cycle the crane either handles only one type of request. This type of separate cycles are performed when the crane has only a small number of tasks in queue, or all of the tasks are of one type. A more common way for the crane to operate is to perform combined cycles, where both storing and retrieval is done in the same cycle. The dual-shuttle cranes in the installed system can handle up to four totes per cycle, two storing tasks and two retrieval tasks. Such a cycle will be called a *quadruple command cycle* (or quadruple cycle) in this work. Because the rack is double-deep, up to six rack locations, other than the I/O point, can be visited in one cycle. A quadruple cycle with two storing, one rearrangement, and two retrieval tasks, is presented in Figure 3. Five different rack locations are visited in the example cycle and the number of crane movements is 7.

The possibility of performing combined cycles depends on the availability of storing and retrieval requests. If both types are available, combined cycles give an advantage with respect to total travel time because the time for performing a combined cycle is always no greater than the sum of single storage and retrieval cycles. [14] In other words, operating the crane so that it handles the maximum amount of totes per cycle, whenever available, leads to the highest throughput. This operating scheme minimizes empty travel of the crane and maximizes its tote handling capacity.

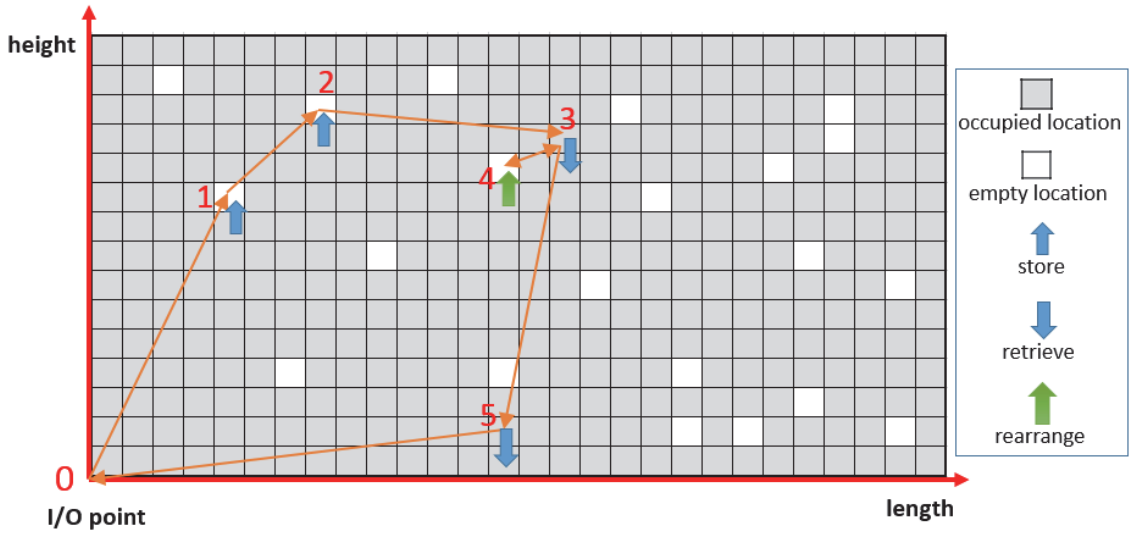


Figure 3: Example of a quadruple command cycle in a double-deep rack

The cycle time consists of two components: travel time and handling time. Travel time is the sum of the time it takes for the crane to move the path between all of the rack locations in the cycle. Handling time is the time the crane needs for extracting or depositing one or two totes. This time depends on the crane's characteristic and can usually be assumed constant. Another constant is the time it takes to accurately position the shuttle in front of a storage location after each movement. This time adds to the travel time of each movement. [15] The more different locations are visited in a cycle, the higher is the proportion of the cycle time which goes to handling the totes. Some additional communication time might also be needed for messaging between the WCS (warehouse control system) and the cranes. In this work the communication delay is not explicitly modeled, but it is taken into account in the allowed calculation times.

## 2.3 Physical design

Typically when designing an AS/RS system, the required number of storage locations $L$ is known beforehand. If the racks are double-deep, as in the installed system, this means $L$ is constant in the equation

$$L = 4 * r * c * n \tag{1}$$

, where $n$ is the number of aisles, $r$ is the number of rows in a rack and $c$ the number of columns [1]. Choosing more aisles reduces rack length and / or height if the capacity is maintained. Shortening or lowering the aisles can reduce storing and retrieving times, because the traveling distances are shorter. On the other hand, the cost of the system is very sensitive to the number of aisles, since each aisle generally needs its own crane. One motivation for designing efficient crane control policies is to enable longer and higher aisles while maintaining the required performance. In some cases, efficient crane control might make the marginal difference, which could eliminate a whole aisle during the

design phase, leading to substantial cost savings. At the same time with the choice of the number of aisles, one has to consider the trade-off between rack height and length. These are influenced by the dimensions of the warehouse building.

In this thesis, a constant acceleration/deceleration model is used to approximate the kinematics of the crane. This gives two possible speed profiles for any horizontal or vertical movement.

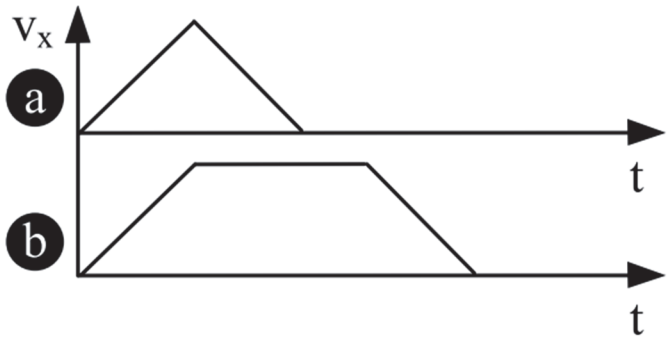


Figure 4: Speed profiles of crane movement with constant acceleration [9]

If the top speed of the crane is $v$ and the acceleration $a$, then $t_s$, the time to travel a distance $s$, can be derived from the laws of constant motion for constant acceleration.

$$t_s = \begin{cases} 2\sqrt{\frac{s}{a}} & \text{, when } s \leq \frac{v^2}{a} \\ \frac{s}{v} + \frac{v}{a} & \text{, else} \end{cases} \tag{2}$$

The upper value corresponds to case a) in Figure 4, where the crane does not reach full speed. Respectively, the lower value is equivalent to case b), where the crane reaches full speed before decelerating. Usually the horizontal and vertical speeds are different. Because the crane moves simultaneously along the x and y axes, the actual travel time $T$ is the maximum of the horizontal and vertical travel times. This is known as the Chebyshev distance metric [1].

$$T = \max(t_x, t_y) \tag{3}$$

A rack where the travel time to the farthest column equals the travel time to the highest row is called square-in-time. This is a common configuration since it minimizes the expected travel time from the I/O-point to an arbitrary rack location. However, due to spatial and mechanical constraints, it is frequently not the chosen design. All racks which are not square-in-time are called rectangular. [14] If $t_x$ is the maximum horizontal travel time and $t_y$ the maximum vertical travel time, then the shape of the rack can be described by a "shape factor" $b$, which is defined as [14]:

$$b = \min\left\{\frac{t_x}{T}, \frac{t_y}{T}\right\} \qquad , (0 \leq b \leq 1) \qquad (4)$$

By choosing $T = 1$, the rack is said to be *normalized* in time [16]. For example, if $t_x$ is longer than $t_y$, the normalized horizontal travel time is 1 and the vertical travel time $b$. When $b = 1$, the rack is square-in-time. The shape factor is a frequently mentioned parameter in AS/RS literature. The effects of control policies are commonly discussed with respect to different values of $b$. The shape factor of the installed system is 0.47, so the maximum vertical travel time is just under a half of the maximum horizontal travel time.

## 2.4 Performance evaluation

**Performance metrics**

In order to evaluate different control rules, it is important to define the performance metrics of interest. The most commonly used performance measure of an AS/RS is system throughput, which is defined as the number of storage and retrieval requests performed by the system per time period [6]. In practice, this means summing up the number of totes handled in all cycles performed over a time period. In the system design phase, it is important to estimate the maximum system throughput which is influenced by the system configuration, the physical design parameters and the crane control policies. Experimentally, the maximum throughput can be measured from the total time it takes to handle a predefined amount of storage and retrieval requests in queue. The maximum throughput for a single aisle is the inverse of the mean expected cycle time [5][16].

In AS/RS study it is frequently assumed that the maximum system throughput of a multi-aisle system is a multiple of single aisle throughput [7]. In practice, the system throughput also depends on load balancing between the aisles, which is a higher level problem managed by the WCS. One main factor in this problem is the allocation of SKUs and their stock in different aisles. Methods for load balancing are not studied in the thesis. It is also important to note that this work considers AS/RS performance independently from other material handling systems and their restrictions. In the whole automatic storage system the throughput of the AS/RS is also dependent on the other parts of the installation. If the picking stations had a lower throughput and the conveyor loop would be too small to compensate for this, then the AS/RS control might have to delay some tasks. Here the buffers are assumed large enough so that these external factors can be dismissed from the control decisions.

Another important measure of performance is the *request turnaround time*, also known as the system response time, which is defined as the time lapse between the arrival and completion of a request. This includes waiting in queue for the crane to arrive, and travel with the crane to the I/O point in case of a retrieval task, or storing location in case of a storing task. [13] In addition to the average response time, the maximum or worst case response time is often considered. A part of the request turnaround time is the *crane*

*response time*. This is the time that it takes for the crane to reach the request from its current location. Retrieval tasks are usually urgent. They can have strict due times, which have to be met. For example a warehouse picking order has a cut-off time, which determines whether the order will make it in time for distribution. In flexible manufacturing applications a production line could stop, if the right component is not retrieved for processing in time. Thus, the earliness or tardiness of performing a request can also be a vital performance metric.

Also an important measure in practice is the number of totes waiting to be stored. If storing tasks are not handled effectively enough, the system can become congested. In a worst case scenario this could lead to a stop of material flow (deadlock) on the conveyors.

**Evaluation methods**

AS/RS research mainly uses two approaches to evaluate the performance of a given system: static travel time models and discrete-event simulation. Static travel time models focus on the steady state behavior of an AS/RS. They use pure mathematical analysis to compute crane cycle times under very specific conditions. All travel time models apply to strictly one system configuration and set of control policies. [5] They normally assume one or more of the following simplifications: continuous and/or square-in-time racks, FCFS sequencing for both storage and retrieval requests, random storage, idle crane positioning at the I/O point. A travel time model also has to use statistics to estimate the relative amount of different types of operating cycles.

Accurate travel time models have been developed for the most common AS/RS configurations. There are also at least two standards by FEM (European Federation of Material Handling) and MHI (Material Handling Institute) which have been issued for calculating cycle times [6][9]. These standards are meant to provide a unified approach to obtain approximate cycle times for different system types. They aim to be simple rather than accurate because their primary use is in contract negotiations [17]. Thus, they should not be compared with more sophisticated travel time models. The research of travel time models is far from exhaustive, because there are so many combinations of operating and design parameters.

Real-world systems are often too complex to be evaluated analytically. One reason for this is that the design and control decisions in AS/RS are linked to some extent. For example, the amount of load handling devices or the depth of the storage rack greatly influence the sequencing problem, and the optimal dwell point depends on the rack shape and the storage policy. These interactions make it difficult to use an analytical approach for evaluation of design and control decisions.

Simulation is a numerical analysis technique designed to evaluate the responses of complex models. It has been used in multiple AS/RS studies over the past 30 years [5] [18]. It is mandatory to use simulation to adequately model all operational features of an AS/RS in its dynamic environment [8]. In a simulation model, it is possible to change the system state and workload of the crane with a set of parameters. Also controls can be

parametrized which makes their comparison feasible. The performance evaluation in this thesis is carried out with the implemented simulation tool.

# 3 Crane control policies

When an AS/RS has been physically implemented, the limits of its operating capabilities are fixed. However, achieving the full potential of the system depends on the way the system is controlled. Crane control policies are the rules which determine the actions performed by the cranes of the AS/RS [8]. The operation of the system is governed by a coherent set of these control policies, each handling a specific subset of activities [1]. Crane control methods try to utilize the available information about current tasks, SKU features and estimated tote flow rates to improve performance. The significance of crane control is hard to assess without experimentation, because it is unique for each system [15].

Generally AS/RS control problems are difficult because, as with other logistics processes, demand is stochastic [19]. Requests are received with short notice, they are hard to predict, and the state of the system changes fast. For example the empty locations in the rack at time $t_0$ are influenced by previously performed storing and retrieval tasks. This is also true for the set of locations containing a certain product. Partially due to these complexities, approaches that focus on formulating robust guidelines are often favored instead of methods seeking an optimal solution [17][19]. Other reasons for an AS/RS supplier to favor these simple approaches can relate to cost efficiency. It is easier to develop and maintain standard control methods using simple heuristics than to develop a complex customized control for each unique installation. [20]

The operation of a crane can involve different objectives and service level constraints. Usually the main objectives are to maximize aisle throughput and minimize request turnaround times. If there are requests with due dates, then minimizing tardiness of those requests should also be taken into account. Sometimes task priorities are also assigned in an upper system which adds constraints to the sequence in which the crane tasks should be performed. The AS/RS should perform robustly as system state parameters such as rack fill level and crane utilization change. This chapter goes over the most important crane control decisions. The crane control policies of the installed system will also be presented.

## 3.1 Storage location assignment

Storage location assignment is a set of rules that determines where incoming storage totes will be located in the storage rack [2]. This is important since storing decisions directly affect expected travel times to future retrieval tasks. A shared storage assignment policy means that there are multiple allowed options to store incoming totes. Whenever such a policy is used, the open location choice is also a part of the cycle formation problem. In addition to affecting travel times, a storage assignment policy also has an impact on the space requirements of the rack, which indirectly affects cycle times. Three storage assignment policies are considered in the following subsections: random storage, full-turnover storage and class-based storage. Because there was no available information

about future SKUs during the making of this thesis, the control rules that are formulated in subsequent chapters of this thesis will assume, that the random storage policy is used. Nevertheless, the feasibility of alternative policies will be assessed in the following sub-sections, based on literature and expert interviews.

**Random storage**

The supplier uses the random storage in the installed system. Random storage is a completely shared storage policy where all incoming storage totes can be stored in any aisle and any open location in the rack. Also, no grouping of totes with the same SKU is usually made. This makes random storage the easiest storing strategy to apply, since it makes no distinction between storage totes based on SKU features. When a tote is retrieved from for picking, it can be returned to a different rack location or even a different aisle. [18] Despite its name, the open locations in random storage are chosen according to an *open location selection rule* instead of a random choice. One basic rule is the *closest open location* (COL) heuristic. This means that an incoming tote is stored to the closest unoccupied location from the I/O point, with respect to travel time. [18] [21] With double-deep racks, the COL rule also needs to decide, whether to allow storing totes in front of other ones, if there are locations with two open positions available. This decision also includes combining two storing tasks in the same double-deep location. Storing totes to front positions forces the crane to perform rearranging movements, which increases retrieval times. These rearrangements could be avoided up to a 50 % fill level. In this thesis only normal production rack fill levels (> 50 %) will be considered, so rearrangements won't be avoided. The COL rule is used in the installed system with some additional priority rules. One of these rules is for keeping both rack sides approximately equally filled. Another modification is made for very high fill levels (> 85 %) to maintain some open locations also in the front of the rack for forced rearrangements. When a shared storage policy is used, open location selection is a part of the sequencing and cycle formation problem. It will be further discussed in section 3.3.

Random storage is the most effective policy in terms of rack space utilization. It is generally a good choice for double-deep racks, because it is difficult to prevent items from mixing due to rearrangements [17]. In the installed system this is emphasized by the use of compartmented totes. Ignoring SKU features has been found to increase expected tote retrieval times. Thus it is the main drawback of random storage. It is equally likely for slow moving items to get stored closer to the I/O point than fast movers. Also, slow moving items can end up being stored in front of fast movers. However it is a false assumption that storage policies which take SKU features into account, always lead to lower expected retrieval times. This topic will be discussed in the following sub-sections.

**Full-turnover storage**

Full-turnover storage exploits the demand frequencies of SKUs in assigning their storage locations. The basic idea is to store fast moving items closer to the I/O point to reduce crane travel times. The demand frequency, also known as the turnover rate, is defined for

each SKU as the number of transactions, both storing and retrieval, during a time period. In full-turnover storage the SKUs are ranked in a descending order according to their turnover rates, and assigned sequentially to the locations that have the smallest retrieval time cost.

In a strict version of full-turnover storage, the storage locations are dedicated, meaning that each SKU is assigned a number of locations in the rack, where only that SKU can be stored. This is referred to as *dedicated storage*. [21] When dedicated storage is used, the COI (cube-per-order index) rule is a well-known ranking method, which takes the space requirements of the SKUs into account. The COI is defined as the ratio of the number of storage locations assigned (or calculated) to an item, to its turnover rate. With this measure, the SKUs with the smallest COI are positioned closest to the I/O point. If each SKU is stored in only one location per aisle, the COI is the reciprocal of the turnover rate. [21][22] Dedicated storage can be problematic, since the locations need to be allocated according to the maximum space requirement of each SKU. The assigned locations need to be reserved even when an SKU is out of stock. These requirements increase the needed storage space. [1][21] For example, random storage needs about 70 % of the space requirement of dedicated storage. This result is based on the assumptions that the changes in inventory levels of different SKUs are independent, and most of the time the space requirement of an SKU is less than its maximum inventory. [21] Another difficulty with dedicated storage is that demand frequencies change constantly as do the SKUs in storage [1]. Because of these changes, the storage rack is never in a perfect full-turnover state for long, because this would require constant repositioning of SKUs.

In some applications with double-deep racks, it is possible to decrease the amount of rearrangements with dedicated storage by allowing only the same SKU to be stored in both front and back positions [20]. In the installed system this strategy could not directly be applied because of the compartmented totes. Also it is expected that there will be several thousand SKUs in one aisle which makes it very difficult maintain the integrity of a dedicated storage location allocation.

An alternative approach to dedicated storage is to calculate a new full-turnover based storage allocation once in a defined time period and to make some *healing* location changes to those SKUs, whose current locations deviate most from the calculated allocation [22]. This relaxation also breaks the requirement of dedicated storage. When a storage tote is fed into storage, or returns from picking, it is preferably stored to its calculated spot, but can also be stored in locations, e.g. the closest open location to its calculated place. This strategy could also be applied for double-deep racks. One problem in storage healing is that the relocations cause extra crane movement which takes away crane capacity from actual production. Especially in periods of high utilization, the crane should not be performing any moves which are not related to production [17].

If the storage rack is single-deep, the retrieval cost of a location is simply the crane travel time from the I/O point to the location, which makes it is straight-forward to rank the locations. With double-deep racks, the retrieval cost of a back position becomes much

higher when another tote is stored in front of it. The cost of rearranging depends on the current open locations in the rack. Rearranging requires at least one extra movement and load handling time. Hence, it can be faster to retrieve a non-blocked tote from the far end of the rack than a blocked tote near the I/O point. In an ideal situation, the full-turnover storage with double-deep racks would require that slow moving items were stored in back positions and fast moving ones in front of them. This kind of order would be hard to achieve and retain because the order in which the storage rack is filled might not be possible to choose, and replenishments of SKUs arrive independently.

Because the whole idea of full-turnover storage is based on the turnover rates of the items, there should be a sufficient amount of sales data and forecasts for the SKUs in storage in order to achieve the advantages. If the calculated turnover rates are inaccurate or unreliable, full-turnover storage should not be considered. [20]

**Class-based storage**

Class-based storage partitions all products into two or more classes and reserves a block of storage locations within the rack for each class. The class partition is based on some criterion, for example COI, duration of stay or turnover rate. It can also be based on the affinity of items, meaning that items which have a higher chance of getting picked in the same order get stored close to each other. This can be problematic though, since the items for the same order are not necessarily retrieved sequentially [22]. If the turnover rate is used, the items with the highest turnover rate are allocated to the class whose storage zone is closest to the I/O point. Inside the zone item locations are chosen according to an open location selection rule. [4] Thus, class-based storage is a combination of full-turnover and random storage policies. The goal of using product classes is to achieve the potential effectiveness of full-turnover based storage while maintaining a part of the flexibility of random storage [1]. A higher number of classes can potentially yield larger travel time savings, but also increases the needed storage space. For a single-deep rack it has been studied, that most of the gain from full-turnover storage can be obtained by using a small amount of classes. For example 96 % of the potential improvement can be achieved with 6 classes, and over 99 % with 12 classes [23]. Using many classes can be hard to manage. In practice, the number of classes is usually 2 – 3 [18] [21]. After deciding the number of classes, there is the problem of deciding, how many and which SKUs belong to which classes. Zoning, i.e. the division of the storage rack into different zones, poses three further problems:

- the shape of the zones
- the size of the zones
- the location of the zones

Zone sizing is commonly chosen based on an ABC-analysis. In case of two classes, the A class is reserved 20 % and the B class 80 % of rack area. With three classes, the sizes for A, B and C classes are 20%, 40% and 40 %, respectively. [2] The classes are usually

L-shaped or rectangular. In Figure 5, a typical example of zone division is presented for three classes.

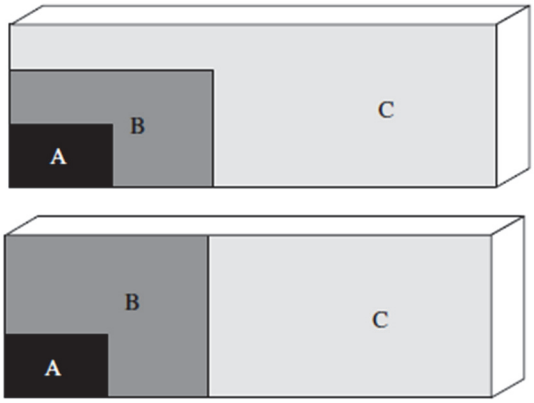


Figure 5: A typical zone positioning for three classes in a square-in-time rack (upper part) and rectangular rack (lower part)[1]

With double-deep racks, zoning is not as simple. If the zones were allocated according to Figure 5, each zone would require its own rearrangement area. The fill level of each zone would need to be controlled so that there would always be a chance to move a tote in an allowed location. Even so, the zone positioning of Figure 5 might not work well double-deep rack. This is due to the previously mentioned issue that the back positions have a higher retrieval costs than front positions. Instead of storing fast moving items in front of other fast movers, it could be beneficial to have a class of slow moving items stored in back positions. Because the order of SKUs arriving to the storage can't normally be controlled, this type of zoning would be difficult to implement and maintain.

The implementation of a class-based storage policy requires a lot of parameters to make all the decisions listed above. The values for these parameters should be carefully chosen based on real production data. Otherwise, applying the class-based storing strategy might not bring any advantages compared to random storing [17].

**Performance of storage policies**

There are three important factors, which have been found to affect the performance of storage assignment strategies: the fill level of the rack, the demand variation of tote flow, and the *locations-to-product ratio* (LTPR). [21][24] The LTPR is defined as the ratio of the total number of occupied rack locations to the number of SKUs in storage. A particularly low rack fill level can be advantageous to random storage with the COL rule, because the COL strategy leads to filling around the I/O point, and empty locations in the back, if there is overcapacity in the storage area. However, the AS/R system should be dimensioned so that fill level of the system stays relatively high during most of the operating time.

The ABC curve of tote flow can be used to approximately describe the variation in the storing and retrieval demands within a group of items. This variation directly influences

the effect that different storage policies have on average retrieval times. The ABC curve is usually presented with two percentages, e.g. a 20% / 80 % ABC curve means that 20 % of the stored items account for 80 % of all retrievals. The ABC curve can also be defined by the function

$$G(i) = i^s \ , \qquad 0 < s \leq 1 \tag{5}$$

, where $G(i)$ is the ranked cumulative percentage of demand versus the proportion of inventoried items $i$. [24]

The demand frequencies $f_i$ for the items can be calculated with the formula:

$$f_i = f_{tot} \left[ \left( \frac{i}{N} \right)^s - \left( \frac{i-1}{N} \right)^s \right] \tag{6}$$

, where $f_{tot}$ is the total demand frequency and $N$ is the total number of SKUs [21]. Figure 6 shows an empirical ABC curve, where $s = 0.4$. This translates to a 20 % / 52 % curve.

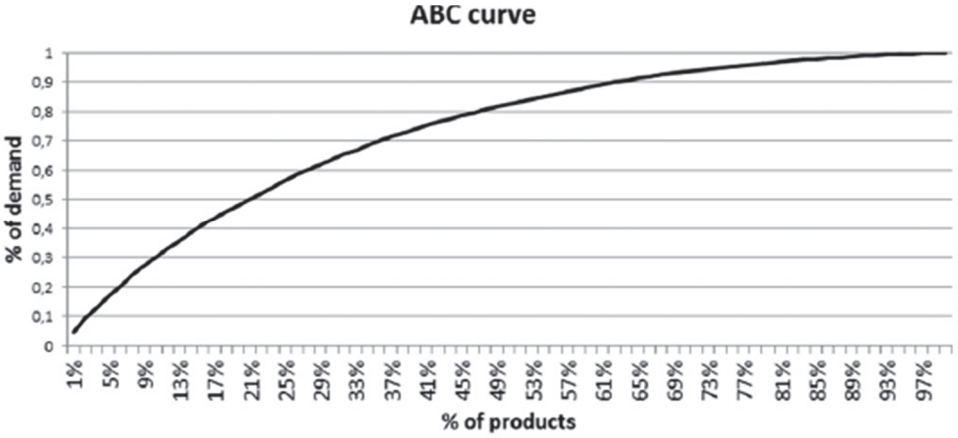


Figure 6: An empirical ABC-curve [21]

Full-turnover dedicated storage has been compared with random storage under different values of the fit parameter $s$. The conclusion of one study was that full-turnover storage has an advantage in retrieval times until the case of approximately a 10% / 33% ABC curve, which translates to $s = 0.5$. With a less skewed (more equal) demand distribution, i.e. $s > 0.5$, the space reduction associated with random storage was found to offset the retrieval time advantage of full-turnover storage leading to both space and retrieval cost advantages. The study in question was focused on a single-deep rack. [21] Using compartmented totes can make the demand frequency distribution between the storage totes more even if multiple low demand SKUs are stored in the same tote. This would increase the demand frequency of that tote, possibly equal that of a fast moving SKU in a single compartment tote. While the assumption that low demand SKUs get stored together is not necessarily true, the use of compartmented totes obscures the demand frequencies of the totes making it hard to use a turnover rate based ranking.

Another important factor on the effect of a storage policy is the LTPR. When LTPR = 1, each product is assigned to only one location in an aisle. Under this assumption, the advantage of full-turnover storage and class-based storage to random storage is more significant. When the LTPR increases, and an item can always be retrieved from any of its locations, the advantage of full-turnover storage is lost. This is significant because LTPR > 1 in many real-life applications [20][24]. In fact, the number of locations assigned to SKUs often follows an ABC curve similar to the demand curve [15].

If LTPR > 1 and it is always allowed to choose any tote with the requested SKU, this might lead to a situation, where totes at the far end of the rack remain untouched for a long time because of higher retrieval times. To avoid this, different prioritizing criteria can be applied to determine which tote to retrieve. These may include one or more of the following: batch number, earliest expiry date, FIFO (first-in-first-out), smallest number of pieces in a tote. The filters in the installed system allocate each retrieval task to exactly one storage tote. Consequently, it is assumed in this thesis that a retrieval task is always fixed to one rack location. This also means that control-wise the LTPR of the system is 1, because totes with the same item are regarded as different.

## 3.2 Dwell point selection

The dwell point is the position where the crane is placed when becoming idle. A crane becomes idle when it has performed all currently known storing and retrieval tasks in queue. The idea of dwell point selection is to reduce the expected travel time to the position of the next storing or retrieval task after the idle period. This can have an effect on the request turnaround times of the system. In studies of dwell point positioning, it is usually assumed that no cost occurs for unloaded crane travel to the dwell point during the idle period, because this travel is not related to any transaction demand. [26] If the energy consumption of the crane is considered, this assumption might have to be reevaluated.

There are four well known strategies for dwell point positioning [25]:

1. Crane stays at the last point it visits and waits there for the next request
2. Crane always returns to the I/O location when becomes idle
3. Crane travels to the gravity center of the rack when it becomes idle. The gravity center means the location which minimizes the expected travel time to a retrieval task.
4. Crane travels to the rack location, which minimizes the expected travel time or the expected maximum travel time to the next request, which can be either a storing task or a retrieval task. This is called the optimal dwell point strategy.

In this thesis the strategies 1, 2 and 4 will be simulated. Strategy 3 will not be considered because it is a simplification of the optimal dwell point strategy 4. The first three are static dwell point strategies, where the crane always travels to the same location when becoming

idle. These strategies are easy to apply because they don't take information about the request profile or rack filling into account. The supplier uses option 1 for the installed system, where the crane stays at its last position if it runs out of tasks. This method is a common choice in many installations [17]. The upside of this "do nothing" policy is that it causes no additional crane movement that is not directly related to a storing or retrieval task.

The optimal dwell point strategy is a dynamic policy where the dwell point is chosen according to the estimated proportions of storing and retrieval tasks [25]. The task proportions can be represented by a parameter $p_s$, whose value is the probability that the first task after an idle period is a storing task. The effects of a dynamic dwell point strategy can best be exploited in manufacturing applications, where the production schedule is known and the retrievals of a certain group of products are expected to be stored or retrieved during a time period. In such applications the short term probabilities of a storing request can be estimated with help of a production schedule. [27] In distribution center projects such as the installed system, it is more difficult to apply a dynamic dwell point strategy, since the patterns of production are stochastic.

**Optimal dwell point for random storage**

If random storage is used and it were known that the first transaction after an idle period is always a retrieval, i.e. $p_s = 0$, then the optimal dwell point would be at the gravity-center of the rack. It has also been shown that the optimal dwell point is at the I/O point if the probability of the next request being a storage is larger than or equal to 0.5. If the probability of a storing request is larger than 0 but less than 0.5 the optimal dwell point lies somewhere between the I/O point and the gravity center. [28] Closed form solutions for optimal dwell point strategies have been developed for square-in-time racks with both random and full-turnover dedicated storage policies, and rectangular racks with random storage [1]. For a normalized rectangular rack with random storage, the efficient dwell-line shown in Figure 7 is given by [26]:

$$y = \begin{cases} x & 0 \le x \le b/2 \\ b/2 & b/2 < x \le 1/2 \end{cases} \tag{7}$$

When $p_s$ is known, the optimal dwell point $(x^*, y^*)$ is the location on the efficient dwell line which satisfies

$$d^* = \begin{cases} \dfrac{1-2p_s}{2(1-p_s)} & 0 \le p_s < \dfrac{1-b}{2-b} \\ \sqrt{\dfrac{b(1-2p_s)}{4(1-p_s)}} & \dfrac{1-b}{2-b} \le p_s < 1/2 \\ 0 & 1/2 \le p_s < 1 \end{cases} \tag{8}$$

, where

$$d^* = \max(t_{x^*}, t_{y^*}) \tag{9}$$

In (9) $t_{x^*}$ and $t_{y^*}$ are the horizontal and vertical travel times from the I/O point to the optimal dwell point $(x^*, y^*)$. These formulas assume that the rack is uniformly distributed, i.e. that the probability of a retrieval request is equally large for each storage location. They also assume that the I/O point is in the lower corner of the rack at point (0, 0). [26] The efficient dwell line of a normalized rectangular rack with random storage is shown in Figure 7. The gravity-center $(1/2, b/2)$ corresponds to $p_s = 0$, position $(b/2, b/2)$ corresponds to $p_s = \frac{1-b}{2-b}$ , and position $(0,0)$ corresponds to $1/2 \leq p_s \leq 1$. [28] If a turnover-based storing strategy were used, the optimal dwell point would be closer to the I/O point than with random storage because the gravity center of the rack would not be at its physical center, but closer to the I/O point.

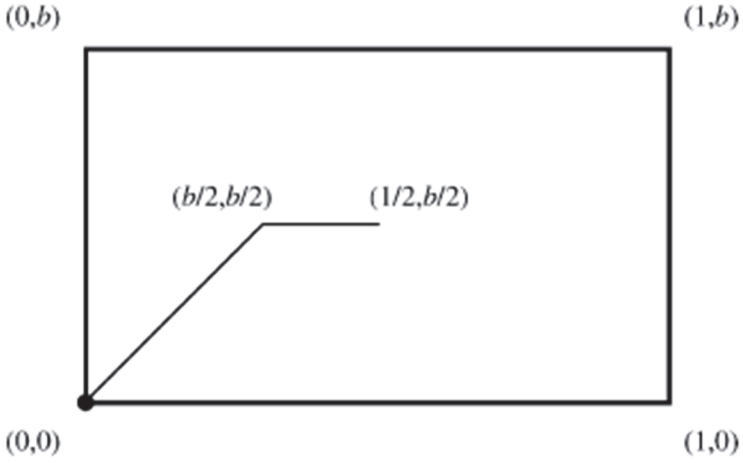


Figure 7: Optimal dwell point locations for a normalized rectangular rack with random storage [28]

If the rate of retrieval tasks and external storing tasks is assumed to be known, the optimal dwell point strategy can be calculated for the installed system. Although this assumption would probably not hold in most production situations, it will made so that the potential benefit of utilizing the optimal dwell point strategy can be evaluated. Because the storage totes recirculate back from the picking stations, the probability of the next crane request being a storing task depends on the amount of totes currently in picking. Let $n$ be the number of totes currently in picking. The picking times can be modeled as independent identically Poisson-distributed random variables with the same rates $\lambda_p$, where $\lambda_p$ is the reciprocal of the expected picking time. Then the returning tote rate at that moment is $n\lambda_p$, a multiple of the individual arrival rates. Let the known rates of retrieval tasks and external storing tasks be $\lambda_r$ and $\lambda_s$, respectively. If there are $n$ totes in picking, and $p_e$ is the probability of a storage tote becoming empty during picking, then the probability of the next request being a storing task is given by:

$$p_s = \frac{(1-p_e)n\lambda_p + \lambda_s}{(1-p_e)n\lambda_p + \lambda_s + \lambda_r} \tag{10}$$

As the number of totes in picking increases, so does the value of $p_s$, moving the optimal dwell point closer to the I/O point.

**Efficiency of dwell point strategies in literature**

Dwell point strategies for a unit-load system have been researched with simulation [25][27]. The average turnover time for a request was used as the performance measure. The studies concluded that the optimal dwell point strategy results to the lowest average turnover time under almost all circumstances. This result is based on the assumption that the probabilities of the type of the next request after an idle period are known. It was also found, that the effect of optimizing the dwell point position was higher with full-turnover storage than with random storage. When random storage was applied, none of the studied methods were dominant compared to the other. [27] The effect of dwell point positioning was also found higher for more rectangular racks, i.e. a low shape factor $b$.[25] [27]

As can be expected, the effect of a dwell point strategy was found significant only in situations, where the frequency of applying the dwell point strategy is high. This translates to a demand profile, where storage and retrieval requests arrive independently in small quantities, leaving the crane idle in between. In [25] the authors compared the effect of dwell point strategies under varying system utilization and rack shapes. It was concluded that under periods of low utilization (30 – 50 %) the difference in average request turnover time between the worst and best strategies was around 20 %. During periods of very high crane utilization (> 80 %), the effect of a dwell point strategy was found negligible.

## 3.3 Request sequencing

Request sequencing has been identified as a control decision which can potentially improve AS/RS throughput in situations, where the crane is at maximum utilization and there are many tasks of both type in queue [1][14][24]. There are many possible methods for request sequencing, especially when it is coupled with the cycle formation problem. Although the sequencing and cycle formation problem is affected by the chosen storing strategy, its effect on performance is independent of uncertainties related to SKU attributes. No statistical information or prediction methods are needed for making the control decisions. The sequencing and cycle formation problem for the installed system type can be stated in the following way:

> *Given a set of known storing and retrieval requests, form quadruple command cycles so that the time to execute all of the cycles is minimized.*

Because the travel time of a combined cycle is always less than the sum of performing its tasks in individual cycles, the optimal sequence contains the maximum number of combined cycles [29]. Even though storing requests are usually not as urgent as retrieval requests, it is a good practice to utilize the maximal handling capacity of the crane to

avoid congestion, when there are both types of tasks in queue. The sequencing problem is dynamic, because the sets of storage and retrieval requests change over time as new requests arrive [15]. The incoming storage totes are queued on an infeed conveyor. Consequently, the order of the storing requests can't be changed, so they need to be completed according to the FCFS (first-come-first-serve) principle. On the other hand, retrieval tasks are just electronic messages coming from the WCS, so their order can easily be manipulated [1][11].

**Breakdown of the sequencing and cycle formation problem**

The form of the sequencing problem changes significantly with the system configuration and the storing strategies. A majority of the research in request sequencing has been done for the single-deep single-shuttle system [14][15][29]. The addition of double-deep storage racks and dual-shuttle cranes brings considerable extensions to the problem, which will be discussed in this sub-section. The snapshot sequencing and cycle formation problem, as stated above, can be broken down into four sub-problems, which are presented below:

1. For each known storing request, choose an open location in the rack
2. For each ordered pair of storing requests, choose two retrieval requests to be performed in the same quadruple cycle.
3. For each quadruple cycle, choose the order of the tasks to be performed. This is known as the routing problem.

For double-deep racks, there is an additional sub-problem:

4. For each blocked retrieval request, choose an open location to move the tote that is blocking it.

The first problem is open location selection, which was mentioned in Section 3.1. Open location selection is dependent on the storage assignment policy. In the case of a dedicated storage policy, the locations of storing requests are predetermined, which makes the choice of storage locations unnecessary. If multiple possible locations exist for the storing requests, as in random or class-based storage, open location selection can be included in the sequencing and cycle formation problem. For single-shuttle systems it has been shown that such an integrated approach can yield a higher increase in throughput compared to sequencing with dedicated storage [15]. With double-deep racks, an additional restriction in open location selection is to not store a tote in front of a known retrieval task.

The second problem, where the requests are paired, is called the *grouping problem*. [30] If the amount of storing tasks in queue $|S|$ is larger than the number of retrievals in queue $|R|$, then only $|S| - |R|$ storing requests need to be considered in the problem, because S is an ordered set. In the case of a single-shuttle system, where one storing task is paired with one retrieval task, the grouping problem can be formulated as an *assignment problem*. The assignment problem is a well know linear combinatorial problem. With a

dual-shuttle system, the grouping problem is intertwined with the third problem, which is called the routing problem. The routing problem is like a small traveling salesman problem with the limitation, that the crane can only carry as many totes as the number of shuttles. In a single-shuttle system, there is only one possibility to route the cycle: first store and then retrieve. When performing a quadruple command cycle, the order of the tasks can be either:

- store – store – retrieve – retrieve
- store – retrieve – store – retrieve

Other alternatives are not possible since the crane can carry up to two totes. With double-deep racks, the number of possibilities increase because of rearrangement moves. Each quadruple cycle can have 0-2 rearrangement moves which need to be performed before the blocked retrieval requests can be handled. Double-deep storage also gives the possibility to combine two storing or rearrangement tasks so that they are deposited in the same rack location. Combining two tasks is usually profitable because it saves one crane movement and one load handling time. This will be considered in the algorithms presented in Chapter 4.

Choosing an open location for the forced rearrangements is another part of the problem. This could be done in a manner, which helps minimize the overall cycle times. However, this choice would have to be coupled with the choice for the storage locations to avoid collisions. Also, it obviously depends on the grouping problem.

**Static and dynamic sequencing**

Two main approaches have been suggested to deal with the dynamic nature of the problem: *static sequencing* and *dynamic sequencing*. [15][29] Both of these include a parameter called the sequencing horizon, $h$. The sequencing horizon is the amount of retrieval and storing tasks which are included in the sequencing problem. The maximum size of the sequencing horizon is $|R|$, the number of retrieval tasks in queue. If $h < |R|$, the requests are sequenced in "blocks" of size $h$. [29] In static sequencing, all the sequenced tasks in a block are performed before considering new tasks. The benefit of static sequencing is that it requires less calculation, because the order of already sequenced tasks is not changed. Also, because all scheduled requests in the block are performed, none of the requests can be bypassed for long. The downside of the approach is that new requests which arrive after the sequencing of the block cannot effect the sequence. Also, the locations that become empty during the execution of a block cannot be taken into account. [29] In *dynamic sequencing*, a complete block of requests is sequenced, but only the first cycle is executed. After the cycle is performed, the status of the rack locations is updated and it is checked whether there are more requests waiting. If so, the block is updated by adding a storage and retrieval request, after which it is resequenced. This dynamic approach can also be generalized by adding a parameter called the *frozen horizon* $f$. The frozen horizon is the number of tasks that will be performed

after sequencing before updating the block and resequencing. The value of $f$ can range from 1 to $h$. [9] Dynamic sequencing has an advantage to static sequencing in that it utilizes the new information that comes from updating the block of requests to be sequenced and the set of open locations.

However, using dynamic sequencing does not guarantee that every request in the sequencing queue gets executed in reasonable time. A retrieval request in the far corner of the rack could wait for a long time if new incoming requests keep bypassing it. Therefore, it might be necessary to employ special rules to ensure that a retrieval at the far end of the aisle is not excessively delayed. [8][31] In some situations this can cause a tradeoff between maximizing throughput versus minimizing response times.

In both static and dynamic sequencing, the size of the sequencing horizon is an important parameter. Lengthening the sequencing horizon means taking more requests into account, thus adding more information to the problem. On the other hand, it also increases the size of the problem, which adds computation. For dual-shuttle cranes, it is necessary to have an even number as the length of the sequencing horizon so that quadruple cycles can be performed.

In the installed system, the aisle of a storage tote has to be decided well before it reaches the infeed conveyor of the first aisle. Including storing tasks in the sequencing problem before they arrive at the end of the infeed conveyor could be risky, because a congestion in the conveyor loop might hinder the tote temporarily, and the sequence of the totes could still change. Thus, the length for the sequencing horizon should be equal or less than the capacity of the infeed conveyor and crane pickup location combined. In the installed system, the length of the infeed conveyor is roughly 3.35 meters and the totes arrive short edge (0.4 m) first, so the capacity is 8 totes. The capacity of the pickup location is two totes. Given these physical constraints, the length of the sequencing horizon will be set to 10 in this work. This means that 5 quadruple cycles can be calculated in one sequencing problem.

**Solution methods**

An ideal sequencing algorithm would make all of the decisions of the presented sub-problems optimally. Additionally, the algorithm should be computationally efficient enough, so that computation time wouldn't cause extra delay in operation.

There have been various approaches to solve the sequencing problem. Because the complexity of the problem varies, so do the suggested solution methods. Most of the literature is concerned with the unit-load AS/RS. [14][15][29] Among the multiple solution that have been presented for the unit-load system are two greedy heuristics, which aim to minimize one or more travel time components of a dual cycle. The nearest-neighbor (NN) heuristic is a greedy algorithm which tries to minimize the travel-between time i.e. the time needed to travel between the storing and retrieval location in a dual cycle. The simple NN heuristic been found to perform reasonably well especially with a

dedicated storage policy [23]. The slightly more complex total-travel-time (TT) heuristic tries to minimize the sum of all three travel components in a dual cycle. [9][24] Both of these heuristics can be extended to dual-shuttle systems if routing is fixed or otherwise handled in the algorithm. The two heuristics have been extended to the case where a shared storage policy is used [15]. In the same study, a linear programming model for solving a generalized sequencing problem of unit-load systems was presented. The authors conducted a simulation study, where they measured the total time to execute 1200 dual cycles. They compared their LP formulation with the previously mentioned heuristics (NN and TT) and FCFS. In the study, the LP formulation led up to 45 % saving in total travel time compared to the FCFS sequencing rule. The LP model also performed more robustly than the heuristics, of which TT was found best. [15]

In another research paper, the effects of both static and dynamic sequencing of unit-load cranes with dedicated storage were studied. [23] The grouping problem was formulated an assignment problem which was solved using the Hungarian algorithm. In the related simulation study, the static approach with the Hungarian algorithm reduced the travel-between of the dual cycle up to 45 %, which increased throughput by 9 %. Dynamic sequencing methods were found clearly more efficient than static ones, reducing the average travel-between time 10 – 20 % more than static approaches. These benefits were achieved already with a short (3 – 5) sequencing horizon. In the same study, a heuristic method with $O(n)$ complexity was developed and found to perform nearly as well as the Hungarian algorithm, which has a $O(n^3)$ complexity. [23]

More recently, sequencing of other system configurations have been studied. One of the early papers for dual-shuttle sequencing extends the nearest-neighbor heuristic to a *minimum perimeter* heuristic, which tries to minimize the two dimensional distance between the four requests in a quadruple cycle. The crane in that study was capable of performing a swap, so only three rack locations in addition to the I/O point had to be visited in a quadruple cycle. [7] In another study, the authors compared the efficiency of numerous local search algorithms including *iterated local search*, *stochastic hill climbing*, and *random-restart hill climbing*, to solve the grouping problem for a dual-shuttle crane. Dedicated storage was used and the configuration allowed the sequence of storing requests to be manipulated. Each of the search algorithms was given a budget of 100 000 function evaluations. The routing problem was handled separately by looping through all the options after grouping was done. [30] Evolutionary algorithms have been suggested to solve the sequencing problem for a system with double-deep racks. The example case with this approach dealt with very large request blocks (e.g. 100 requests) and a relatively small rack size. Also, computation times of several minutes were allowed, which suggests that the crane response time was not an important performance indicator. [32]

To the author's best knowledge, sequencing and cycle formation problem for double-deep dual-shuttle cranes hasn't been studied in literature even though there are many suppliers that deliver this system type. The lack of literature could be due to the presumption that

the possibilities to shorten cycle times to increase system throughput are more limited with this configuration. With fast cranes, relatively long handling times and a short optimization horizon, the advantages of more sophisticated sequencing methods might be considered too small for the effort. In the next chapter, alternative solution methods for the sequencing and cycle formation problem of this system type are formulated.

# 4 Algorithms for sequencing and cycle formation

In this chapter, three alternative control methods for the sequencing and cycle formation problem of the double-deep, dual-shuttle AS/RS will be presented and compared. One of these is a replication of the control used by the supplier of the installed system, one is a greedy heuristic, and the third one is a mixed integer linear program (MILP). The two latter algorithms are adaptations of similar algorithms, which have been suggested for the unit-load system. Due to the complexity of the problem and strict calculation time constraints, none of the methods combine all of the aspects of the sequencing problem, but are combinations of rule-based heuristics and local or global optimization. Globally optimizing in this context means optimization over all of the requests which are known and included in the sequencing horizon.

All of the sequencing and cycle formation algorithms have to be computationally efficient. The upper bound of the calculation time should be less than the time to perform the shortest possible quadruple cycle. This is because the smallest value for the frozen horizon is 2 + 2 requests, so the next cycle would in some situations have to be calculated while performing the previous one. This calculation time limit is necessary to ensure, that the crane doesn't stand idle, when there are requests in queue.

The smallest quadruple cycle is such that the crane picks up two totes at the I/O point and stores them in the location next to the IO point and retrieves two totes from the opposite rack location. Therefore, the lower bound for the quadruple cycle time is:

$$T_{lb} = 3h + 2p \tag{11}$$

, where $h$ is the constant load-handling time and $p$ is the crane positioning time which is made after each movement. In the installed system $T_{lb} = 11.7$ seconds. Extra time should also be reserved for the communication of the calculation result, so the actual calculation time should be well under $T_{lb}$.

## 4.1 Supplier's algorithm

The supplier of the installed system uses a combination of the FCFS rule and a nearest-neighbor heuristic to sequence crane tasks. Cycle routing is performed in a fixed order: first store, then make the needed rearrangements, and finally retrieve. The open locations for the two storing tasks are chosen according to the following rules:

1. If there are locations in the rack where there are two open positions, choose the one closest to the I/O point
2. Choose two separate locations, with at least one open position, that are closest to the I/O point
3. If a location with two free positions wasn't found in 1, or it is faster to store the two totes in the separate locations chosen in 2, then the separate locations are chosen. Otherwise, both totes are stored in the location found in 1.

The retrieval tasks are selected and sequenced as follows:

1. The algorithm always selects the oldest retrieval task in queue to be performed in the next cycle.
2. The other retrieval task is selected as the closest retrieval to the position, where the second of the two totes is stored
3. The order in which the two retrievals are executed is chosen based on a travel-time comparison, the fastest sequence is chosen.

If only one rearrangement needs to be made, then it is done directly before the retrieval. If two rearrangements need to be made, then they can be combined. A similar time comparison is made as was done with the storing tasks, which follows the steps below:

1. If there are locations in the rack where there are two open positions, choose the one closest to the second retrieval.
2. Choose two separate locations, the ones closest to the two retrieval tasks
3. If a location with two free positions wasn't found, or it is faster to rearrange the two totes in the separate locations found in 2, then those locations are chosen. Otherwise, both blocking totes are rearranged to the location found in 1.

The algorithm is applied with the smallest possible frozen horizon ($f = 2$), meaning that only one cycle will be performed before the queue is updated and resequenced. With this approach, it is not even necessary to calculate more than one cycle at a time. As one cycle is performed the sets of requests are updated and the next cycle is calculated.

A benefit of always including the oldest request in the next cycle is that no retrieval tasks can get stuck in the sequencing queue for a long time. Another strength is that the algorithm is computationally efficient, $O(n)$ because its implementation doesn't require nested loops. The main weakness of the algorithm is that it forms the cycle in pieces: the problems of open location selection, retrieval task selection and rearrangement location selection are treated separately.

## 4.2 Total travel time heuristic

In this section, a greedy heuristic algorithm for solving the sequencing problem is formulated. It is an extension of the total-travel-time (TT) heuristic for single-shuttle systems[15] [29] so it will be referred to by the same name. This algorithm was chosen as an alternative heuristic to the one which the supplier uses. It takes more information about the travel time components of a cycle into account in forming locally optimal cycles. The single-shuttle version of this heuristic has proven to work well [15].

The following notation will be used:

$S$: The ordered set including all pairs of storing requests within the sequencing horizon. $s \in S$ is a pair of two subsequent storing requests.

$R$: The set of all retrieval requests within the sequencing horizon, each targeting one specific location in the rack. $(j, k) \in R$ is an ordered pair of retrieval requests $j$ and $k$.

$d_j$: A binary variable with value 1, if retrieval request $j$ is blocked by another tote, and 0 if it is not.

$m_j$: Rearrangement location for retrieval request $j$, if it is blocked.

$q_j$: The rack location where retrieval request $j$ is located.

$P$: The set of rack locations with at least one open position. $p_1 \in P$ and $p_2 \in P$ are the open location where the first and second tote of storing request pair $s$ are stored. All open locations are available to all storing requests.

$b$: A binary variable that gets the value 0, if storing locations $p_1$ and $p_2$ are the same, and value 1 otherwise.

$h$: Constant load-handling time, which here is equivalent to the average tote depositing time of the crane.

$D_{p_1 p_2 jk}$ : The distance metric used in (12), which consists of all the travel time components in a quadruple cycle.

A simplified pseudo-code version of the TT heuristic is presented in Figure 8.

**For each** ordered pair of storing requests $s \in S$ in queue **Do**

Choose $p_1 \in P$: the closest open location (COL) in the rack

**For each** open location $p_2 \in P$ **Do**

**For each** ordered pair of retrievals $(j,k) \in R$ **Do**

Select triplet $T$: $(p_1, p_2^*, (j,k)^*)$ which minimizes the distance metric $D$

Send cycle to the crane and update the sets:

$S \leftarrow S - s$

$R \leftarrow R - \{j,k\}$

$P \leftarrow P - \{p_1, p_2\} + \{q_j, q_k\}$

Figure 8: A simplified pseudo-code of the total-travel-time heuristic

The TT heuristic seeks to minimize a distance metric, which consists of all the travel time components in the cycle. The distance metric is:

$$D_{p_1 p_2 jk} = t_{IO,p_1} + b(t_{p_1,p_2} + h) + t_{p_2,j} + d_j 2t_{j,m_j} + t_{j,k} + d_k 2t_{k,m_k} + t_{k,IO} \qquad (12)$$

In (12) the travel time between two locations $i$ and $j$ is marked with $t_{i,j}$. The symbol $IO$ represents the I/O point. As can be seen from the distance metric, the heuristic assumes a routing where both storing tasks are made before retrievals, and possible rearrangements are made individually right before retrieving the next tote. Because it is a greedy heuristic, the TT algorithm forms the cycles sequentially, starting from the fastest quadruple cycle that can be formed from the requests in the sequencing horizon. The TT algorithm has a larger time complexity than the one used by the supplier. This is because the set of retrievals has to be looped twice to go through all the ordered pairs of retrievals, raising the number of nested loops to four, thus resulting in a time complexity of $O(n^4)$. Because the sequencing horizon is limited, this should not cause computational problems with an average PC.

## 4.3 Linear programming model

This section will present a mixed integer linear program (MILP) to solve the sequencing problem for the installed system type. It will be called the LP model in the rest of this work. This formulation is an extension of a similar algorithm which has been proposed for unit-load systems [15]. The LP model is an alternative to the heuristic methods used by the supplier and the TT algorithm in that it aims to minimize the joint time to perform all of the tasks in the sequencing horizon, instead of choosing many locally optimal cycles sequentially.

**Modeling simplifications**

There are some simplifications in the model. They are made partially, because of the difficulty to include all of the sub-problems in the formulation of one optimization problem. Attempting this would possibly lead to an obscure objective function or an excess number of decision variables. Solving the sub-problems separately is also not a good option because they are dependent on each other. Another reason to include simplifications is to reduce the size of the problem to keep the calculation time bounded and low enough. It should be noted that the LP formulation was developed in this thesis, so its computational feasibility was not known beforehand. Computational issues will be further discussed in the end of this section, and actual computation times will be shown in section 5.1.

The first simplification in the model is that the storage location for the first storing request in each ordered pair is chosen beforehand, according to the following rules:

1. Find $l_1$ the closest open location to the I/O point (in travel time). This location may have one or two open positions.
2. If $l_1$ has only one open position, and there are locations in the rack where there are two open positions, find $l_2$, the one closest to the I/O point.
3. If $l_1$ had two open positions, then choose it. If $l_1$ had one open position and $l_2$ was found, choose $l_1$ if the sum of the travel time from the I/O point to $l_1$ and the crane handling time is shorter than travel from the I/O point to $l_2$. Otherwise, choose $l_2$.

The extra load handling time is added to the choice of $l_1$ with one open position because the crane needs to visit a second storing location, which adds to the total time of the cycle. The choice of the predetermined open location could be made by random, or by some other rule. The closest-open-location rule was chosen mainly, because it is simple. Choosing the one storing location heuristically greatly reduces the number of decision variables. If $h$ is the length of the sequencing horizon, then $h/2$ locations are reserved before each calculation of the sequencing problem. A different rack location is chosen for each pair. Those, which have only one available position are reduced from the set of open locations.

Another simplification is that all rearrangement locations for blocked retrieval tasks will be chosen in advance. Namely, the closest open location to the retrieval point is chosen for each rearrangement move. This simplification is made to reduce the amount of decision variables in the problem. These locations are reserved and reduced from the set of open locations before sequencing.

Finally, the order of performing the tasks will be fixed to: store – store – retrieve – retrieve. This means that the crane always stores two totes first, either together or separately, and only afterwards starts retrievals. If rearrangements need to be made, they are made before picking up the tote to be retrieved, after which the crane returns to the

retrieval location. The two retrievals can be performed either order. This routing simplification is essential to the formulation of the cost vector and objective function of the LP model.

**LP model formulation**

The notation used for the LP model mainly follows the notation of section 4.2 with some additions. For the sake of clarity, the full notation is listed below:

$S$: The ordered set including all pairs of storing requests within the sequencing horizon. $s \in S$ is a pair of two subsequent storing requests.

$R$: The set of all retrieval requests within the sequencing horizon, each targeting one specific location in the rack. $(j, k) \in R$ is an ordered pair of retrieval requests $j$ and $k$.

$d_j$: A binary variable with value 1, if retrieval request $j$ is blocked by another tote, and 0 if it is not.

$m_j$: The rearrangement location for retrieval $j$, if it is blocked.

$P$: The set of open locations which can be used to store any for storing the second tote in a cycle, $p \in P$. All open locations are available to all storing requests.

$l_p$: The number of totes that can be stored in open location $p$. $l_p = \{1,2\}$ for all $p \in P$.

$o_s$: The open location which is reserved for the first tote in the storing request pair $s$. Location $o_s$ is a member of $P$ only if it has a second open position.

$h$: Constant crane handling time

$b$: A binary variable that gets the value 0, if locations $o_s$ and $p$ are the same and value 1 otherwise.

$t_{spjk}$ : The cost vector, which represents the time it takes to complete a quadruple command cycle in the following order:

1. The pair $s$ of totes is stored in locations $o_s$ (the pre-selected location) and $p$. Notice, that the two can also be the same if there are two free positions in $o_s$. Because $o_s$ is pre-allocated for each storing task pair, it is not considered in the objective function.
2. The ordered pair of retrieval tasks $(j, k) \in R$ is retrieved from their locations respectively. If rearrangements have to be made, they are made to the closest open locations from the retrieval points.
3. Returning from the second retrieval point to the I/O point.

$x_{spjk}$: A binary variable that gets the value 1, if ordered storing task pair $s$ is combined with retrieval requests $j$ and $k$ to form a quadruple cycle, and the second tote of $s$ is stored in location $p$.

**Objective function:**

$$\hat{x}_{spjk} = \text{Min} \sum_{s \in S} \sum_{p \in P} \sum_{(j,k) \in R} t_{spjk} \; x_{spjk} \tag{13}$$

The objective function (13) minimizes the sum of the travel time to perform all tasks in queue. The cost vector $t_{spjk}$ consists of all the travel components of a quadruple command cycle.

**Cost vector:**

$$t_{spjk} = t_{IO,o_s} + b(t_{o_s,p} + h) + t_{p,j} + d_j 2t_{j,m_j} + t_{j,k} + d_k 2t_{k,m_k} + t_{k,IO} \tag{14}$$

The cost vector is practically the same as the distance metric of the TT algorithm. Notice, that an extra load handling time $h$ is added to the cost if the two storing locations are different.

The constraints of the problem are listed and explained below.

**Constraints:**

$$\sum_{p \in P} \sum_{(j,k) \in R} x_{spjk} = 1 \qquad , \forall \; s \in S \tag{15}$$

Each pair of storing requests is in exactly one cycle, and paired with two different retrieval requests.

$$\sum_{s \in S} \sum_{p \in P} \sum_{j \in R} x_{spj} + \sum_{s \in S} \sum_{p \in P} \sum_{k \in R} x_{spk} = 1 \qquad , \forall \; j \in R \tag{16}$$

$$, \forall \; k \in R$$

$$, j \; \neq k$$

Each retrieval request is included in exactly one cycle. The request can be either the first or the second retrieval in the cycle.

$$\sum_{s \in S} \sum_{(j,k) \in R} x_{spjk} \leq l_p \qquad , \forall \; p \in P \tag{17}$$

Each open location is used for equal or less storing tasks than the current capacity of the open location $l_p$ allows. If there are two free positions in the location, then two storage tasks can be performed there.

$$x_{spjk} \in [0,1] \tag{18}$$

The decision variables are binary.

An additional constraint is that the same storage location cannot be used for both storing and retrieval. This is because the order of the tasks would determine if it is possible. This restriction also prevents a storage tote from being stored in front of a known retrieval. This can be achieved by removing the retrieval locations from the array of open locations before sequencing. The amount of open locations for each pair of storing tasks is always 1. It is possible to choose that both totes are stored in the pre-reserved location (if it has two open positions), but this is not mandatory.

The main limitation of the LP model is that it is static. This means that all the changes in the rack caused by the tasks which are being sequenced are considered to happen simultaneously**.** If for example the storage locations are changed for individual cycles after the sequencing problem is solved, the rest of the cycles would have to be checked or resequenced. Another limitation is that the formulation requires an equal number of storage and retrieval tasks. The length of the sequencing horizon should also be an even number to make it possible to form pairs.

**Computational issues**

The formulation of an MILP is perhaps the most important factor in its computational efficiency [33]. One important measure in the formulation is the size of the problem. Adding more variables increases the solution space and amount of calculation needed. The constraints define the feasible region of the solution space. If the size of the solution space is too large, then it needs to be reduced by adding more constraints, or by reducing the number of decision variables. In the LP model presented above, the number of decision variables grows rapidly as the sequencing horizon is lengthened, because the number of ordered pairs, given by $|R|\ nPr\ 2$, increases rapidly. In this work, the sequencing horizon is fixed to 10 due to the physical constraints of the installed system.

Another important factor on the number of decision variables is the set of open locations. If $h$ is the length of the sequencing horizon, then then number of decision variables $v$ is:

$$v = \frac{h}{2}(h\ nPr\ 2) \tag{19}$$

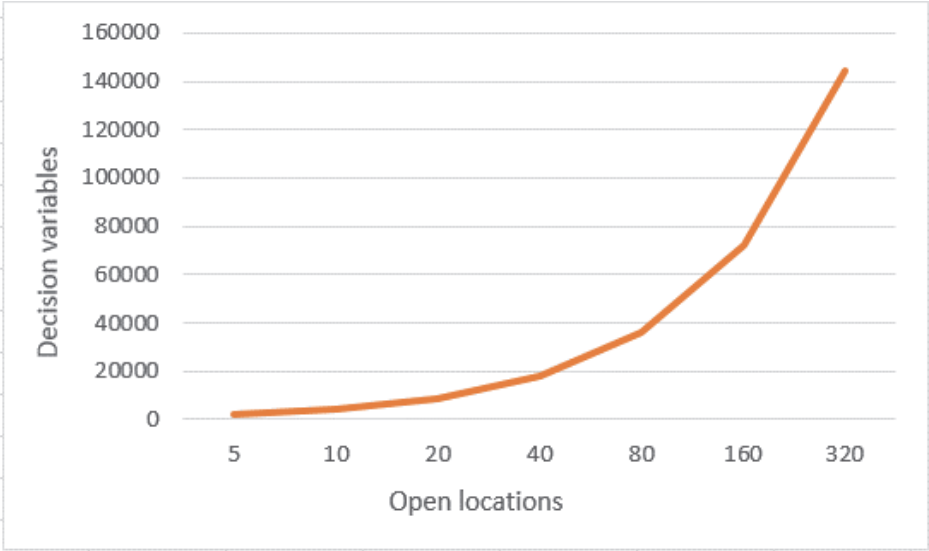


Figure 9: Number of decision variables as a function of the number of open locations with $h = 10$

As presented in Figure 9, the problem size grows fast when more open locations are taken into account. This means that the computational performance of the LP model would possibly deteriorate when the fill level of the system decreases. To keep the size of the problem bounded, an additional parameter will be introduced to limit the number of open locations included in the problem. This parameter will be called *size of the search neighborhood* and denoted by $n$. For example, $n$ =100 means that a maximum of 100 open locations are included in the problem. The open locations in the search neighborhood will be chosen evenly throughout the rack length to give more differing options to choose from. All the locations $o_s$ that were chosen for the first storing task of each pair will be included in the search neighborhood, if they have a second open position. This is done to assure the possibility of storing two totes in the same location.

# 5 Simulation model

A simulation tool was built in this thesis to enable an accurate analysis of AS/RS control decisions of the implemented system. The model was designed with a generic mindset to enable analysis of other system sizes and configurations. The structure of the program is modular; the logic for different configurations and control policies is accessible through a set of parameters, which can easily be changed between simulation runs.

## 5.1 Tools and structure

The parts of the simulation tool, as well as the information flows between them, are presented in Figure 10. The following sub-sections will go through the implementation of the different parts.

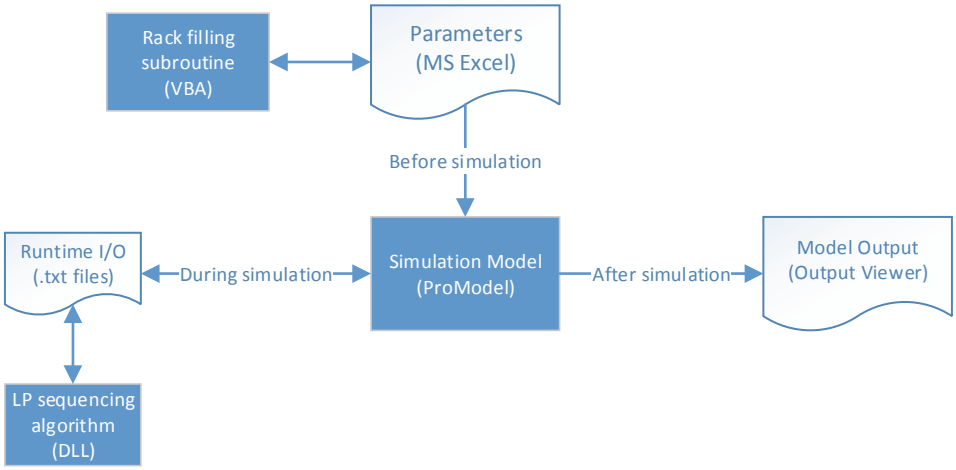


Figure 10: Parts of the simulation tool

**Simulation model**

The simulation model was implemented using ProModel 2014, an object-oriented Windows-based simulation tool. It is the primary simulation tool used at EP-Logistics and well suitable for analyzing discrete part production processes. ProModel has built-in programmable element types such as *locations*, *entities* and *resources*, which are used to create the simulation models. The logic is implemented with a built-in programming language. ProModel can read and write information to and from Microsoft Excel and text files. [34] All the run-time logic of the simulation model was programmed in ProModel, except the LP sequencing algorithm.

The controls for running the model are in ProModel. The animation can be set on or off and the simulation can be paused at any time to check the current state of the parameters and arrays in the model. Some parameter values such as the number of tasks in queue, the current position of the crane, and number of totes it is carrying, are displayed in the UI during runtime. A snapshot of the runtime environment in ProModel is shown in Figure 10.

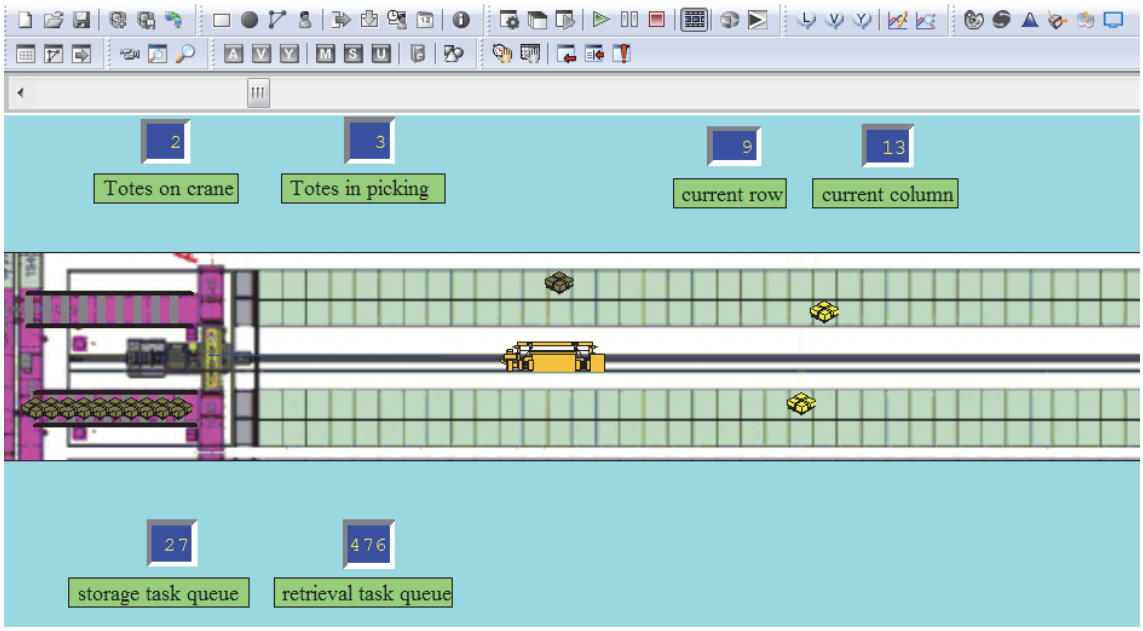


Figure 11: Snapshot of the ProModel UI

**Parameters**

The simulation tool built in this master's thesis can be used to simulate almost all the configurations presented in Table 1. Currently the only limitation is that a crane always operates in one dedicated aisle. There can also be only one I/O point in an aisle, but its location can be changed.

The parameters of the simulation model are read from an Excel spreadsheet file upon starting the simulation in ProModel. The parameters can't be changed during the simulation run. A comprehensive list of the parameters is presented in Figure 12. The parameters can be classified into four groups: tote flow parameters, crane parameters, rack parameters and control parameters. The tote flow parameters define the amount of storage and retrieval requests arriving into the model. This group also has two picking-related parameters: average picking time and tote empty probability, which is the probability of the tote becoming empty during picking and thus not returning to storage. The crane parameters hold all the mechanical parameters for crane operation, such as speeds and tote-handling times. The rack parameters define the dimensions of the storage area, and the number of locations in the rack.

The rack filling subroutine is in the same Excel file as the parameters. It only takes one parameter, which is the desired filling level. The algorithm is iterative. It uses the Rnd() function in Excel to generate random integers between 0 and the depth of the rack, in this case, 2. If the rack fill level is too low after the first iteration, the lower bound of every other random number is incremented to 1 and the new filling is started from the first column. Oppositely, if the fill level is too high, the upper bound of every other random

number is lowered to 1 and the new iteration is started from the back of the rack. The goal of this algorithm is to generate a near real life situation, where there are a little more openings in the back than near the I/O point, but random free locations are available in every part of the rack. Also, the algorithm fills both sides of the rack to the same fill level within a 1 % accuracy so that special balancing rules do not need to be taken into account.

The control parameters are the most important ones for this work. They hold all the crane control options for sequencing, storage assignment and dwell point positioning. In this thesis, only the control parameters and tote flow parameters are studied.

| | Parameter | Value | |
|---|---|---|---|
| Tote flow parameters | Initial retrieval task queue | 0 | |
| | Initial storing task queue | 0 | |
| | Number of retrieval requests per hr | 43,3 | |
| | Number of external storing tasks per hr | 5,25 | |
| | Wave size | 1 | |
| | Average picking time (sec) | 107 | |
| | Single storing task waiting time (sec) | 30 | |
| | Tote empty probability | 0,07 | |
| Crane parameters | Crane Horizontal Velocity (m/min) | 300 | |
| | Crane Vertical Velocity (m/min) | 240 | |
| | Crane Horizontal accel/decel (m/sec2) | 2,5 | |
| | Crane Vertical accel/decel (m/sec2) | 2 | |
| | Crane pickup time (sec) | 3,8 | |
| | Crane deposit time (sec) | 3,5 | |
| | Crane positioning time (sec) | 0,3 | |
| | Number of shuttles (load handling devices) | 2 | |
| Rack parameters | Number of Columns per Rack | 59 | |
| | Number of Rows per Rack | 23 | |
| | Single-deep racks (1) / Double-deep racks (2) | 2 | |
| | I/O point height (row) | 4 | |
| | Storage Cell Height with clearances (m) | 0,3 | |
| | Storage Cell Width with clearances (m) | 0,5 | |
| | Initial fill level (%) | 70 | |
| Control parameters | Sequencing algorithm | 3 | |
| | Sequencing horizon | 10 | |
| | Frozen horizon | 10 | |
| | Size of search neighborhood (only for LP model) | 150 | |
| | Optimization time limit (sec, only for LP model) | 25 | |
| | Dwell point strategy (see comment) | 4 | |
| | Storing strategy | 1 | |
| | ABC curve parameters (x / y) (only for turnover-based storage) | 80 | 20 |

Figure 12: Parameters of the simulation model

**LP sequencing algorithm**

The LP sequencing algorithm was programmed with C++ in Visual Studio and compiled as a Win32 DLL. The simulation model can call the exported functions in the DLL during runtime with the XSUB -function in ProModel. The choice to use an external programming language for the optimization problem was made for two reasons: Although

ProModel provides a very effective simulation engine and the objects to build the simulation logic, it lacks the freedom of using sophisticated data types and structures. Practically all of the information in ProModel is assigned to Integer or Real datatypes, which can be stored in Arrays. Also, an external programming language enables the use of external libraries with preprogrammed solvers for optimization problems, which can save a lot of programming time.

The optimization library used in this work is Google OR-Tools, which is a set of operations research tools written in C++ at Google. OR-Tools is open source and provides an interface to several linear programming and mixed integer programming solvers. [35] The solver which was used to solve the problem is CBC. It is an open-source MILP solver developed by COIN-OR (Computational Infrastructure for Operations Research). The COIN-OR is a project started in 2000, which aims to provide high quality open-source software for optimization as well as other problems in operations research. CBC is a branch-and-cut algorithm. [36]

Branch-and-cut methods are common for solving mixed integer programs. They are a combination of a branch-and-bound algorithm and cutting plane method. [37] Such algorithms work by solving a sequence of linear programming relaxations of the integer programming problem. Branch-and-bound algorithms divide the original problem into smaller sub-problems, or nodes, which are solved sequentially. The use of bounds for the function to be optimized, combined with the value of the current best solution enables the algorithm to rule out some parts of the solution space, meaning that they are searched only implicitly. [37] Cutting plane methods improve the linear programming relaxation of the problem to more closely approximate the integer programming problem. When using solution methods which involve branching, there is a chance that the bounding aspects are not invoked, which can lead to a huge number of sub-problems. The worst case amount of sub-problems for a problem with $n$ binary variables is $2^n$. This exponential growth is possible with all solution algorithms for integer programming, unless $\mathrm{P} = \mathrm{N}$. [33] Solving a mixed integer program is computationally more difficult than solving a normal linear program. Depending on the algorithm used, the solution of an integer program might require solving thousands of linear programs. Therefore the program code used to solve the problem can have a great impact on the calculation time. [33] The CBC algorithm was chosen in this work because it was readily accessible in OR-tools and proved efficient enough for solving the LP model. A summary of average calculation times for solving the LP model is presented in Table 3. Based on these experimentations, 150 locations was chosen to be the size of the search horizon.

Table 3: Calculation times for the LP model with 100 repetitions

| search horizon size (rack locations) | num of variables | avg. solution time (milliseconds) | max. solution time (milliseconds) |
|---|---|---|---|
| 30 | 13500 | 277 | 522 |
| 50 | 22500 | 494 | 915 |
| 100 | 45000 | 1022 | 1584 |
| 150 | 67500 | 1927 | 3277 |
| 200 | 90000 | 2193 | 4060 |

## 5.2 Modeling

This section will give some details about how the racks, requests and crane movements of the AS/RS were modeled in ProModel.

**Routing and processing**

Simulation models in ProModel are built by routing *entities* through locations where they are processed according to a user-defined programmable logic. A simplified flow chart describing the routing of the main entity types in the model is presented in Figure 13.

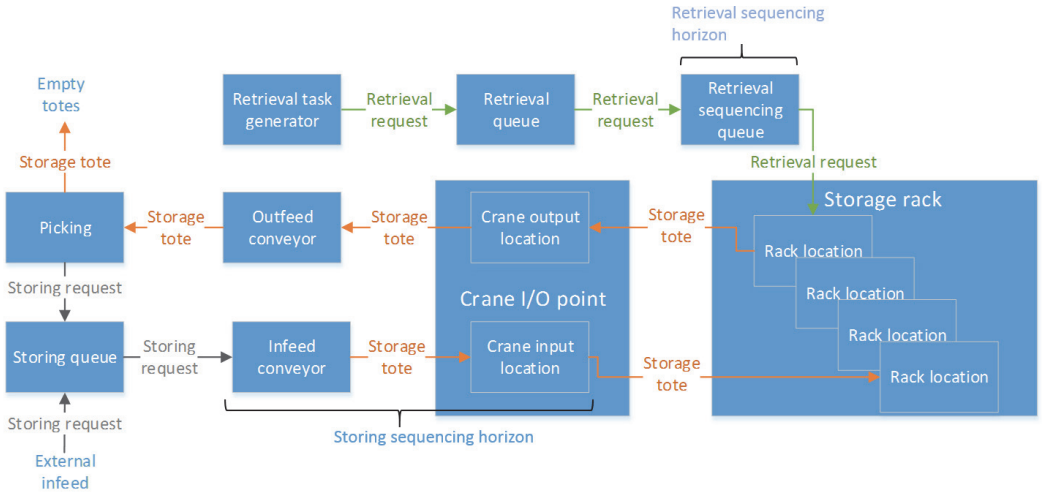


Figure 13: Routing of main entity types in the simulation model

Storage racks are modeled as three-dimensional arrays. Because there is a rack on both sides of an aisle, each storage location in the aisle can be represented by four coordinates (side, row, column, depth). In the model, the racks are filled with generic storage totes; SKUs are not modeled explicitly. The number of totes (0 – 2) in each location is tracked and updated. Each retrieval request can only be fetched from one specific location in the aisle. Thus, they are modeled as 4-tuples with the same coordinates as rack locations. This is in accordance with the installed system, where the stock of an SKU is filtered according to several attributes or rules, including earliest expiry date, batch ID, serial number and FIFO. Because random storage is applied for the scenarios in this thesis, the demand

frequency of each rack location is assumed the same. This can cause a minor modeling error because it is slightly more likely for a request to fall on a location with two totes instead of one. However, the effect of multi-compartmented totes might also obscure this. The full-turnover storing scheme can be simulated in the model, although it is currently implemented without modeling SKUs. Full-turnover storage is mimicked by calculating the demand frequency of each storage location based on the two ABC-curve parameters and generating retrieval tasks according to the obtained distribution.

Upon creating retrieval requests, they are assigned coordinates of a rack location that has at least one tote. Only one request for each tote position is allowed simultaneously. Creating a retrieval request makes a reservation for one tote in the location. Sometimes it might happen, that there are two retrieval requests, which have arrived at different times for the same location. If these tasks are not in the same sequencing horizon, the one for the back position might be executed first. This means the tote in the front position is rearranged and the original location is left empty. In this situation, the request for the front position will be discarded and regenerated for some other occupied location.

The retrieval tasks arrive in an infinitely large retrieval queue, from which they are routed into the sequencing queue in FCFS order. The amount of requests in the sequencing queue is tracked continuously. If it passes a predefined limit, which by default is the length of the sequencing horizon, then the requests are sequenced with the chosen algorithm. When sequencing is initiated, the algorithm checks if there are any storing tasks on the infeed conveyor or the pickup location and includes them in the sequencing problem. Sequencing will be done for the retrieval requests even if there are no storing requests in the infeed area. If there aren't enough retrieval tasks for sequencing, they will be performed in FCFS sequence, so that the crane doesn't have to wait.

All retrieved totes are routed from the outfeed conveyor to a picking location, which is infinitely large. Picking is modeled as a stochastic process. Picking time is distributed exponentially with a rate that is the inverse of the average picking time. Totes can also become empty during picking. Empty totes will disappear from the system after picking instead of returning to storage. The probability for a storage tote to become empty is included in the parameters.

New totes can enter the system from an external infeed source. The arrival frequency can be set with the tote flow parameter "Number of external storing tasks per hour" All arriving storage totes, including the ones that are returning from picking, are first routed to an infinitely large storing queue from which they are fed onto the infeed conveyor. If a sequencing algorithm hasn't chosen an open location for a storage tote by the time it is picked up by the crane, it will be stored to the closest open location. If a storage location does not have another tote stored in it, the incoming tote will always be stored in the back position.

**Crane movement**

Because the two-dimensional animation is from the top-view perspective, only horizontal motion is visualized. Vertical travel is modeled as an additional waiting time when vertical movement time exceeds horizontal movement time. A constant crane positioning time is added to each movement, except when the crane returns to the I/O point. The travel times between two arbitrary rack locations, including the I/O point, were calculated beforehand according to the constant acceleration model in equation 2. The travel times are stored in a matrix $T_{mn}$ in which the element $t_{mn}$ means the time to travel an offset of $m$ rows and $n$ columns. This matrix is kept in the spreadsheet where the parameters are stored and it is updated automatically when crane parameters are changed. Travel times are assumed to be constant and symmetric. The crane's pickup and depositing times are also assumed constant.

The crane can perform all types of cycles with 0 – 2 storing tasks and 0 – 2 retrievals. A quadruple command cycle is performed whenever there are enough requests available. The crane can acknowledge new retrieval requests in the middle of a cycle if it is carrying less than two totes. If only one storage tote arrives at the infeed, the tote is not taken to storage immediately. Instead, the system waits for a possible second tote to arrive at the infeed for a predefined time. The waiting time in the simulation is set to 30 seconds. If a retrieval request comes during the waiting time, it will be served before the storing task. This value is also used in the installed system.

The crane becomes idle, if it finishes a cycle and there are no more tasks in either queue. Even if there is a single storage tote waiting at the infeed for a possible second tote, the crane stays active. If the crane becomes idle, it will always move to the dwell point before serving the next request. This can cause minor response time delays, because the crane may still be traveling to the dwell point, possibly in the wrong direction, when the next request arrives. As a matter of fact, this built-in limitation of ProModel is consistent with actual PLC controls, which usually do not incorporate a function to interrupt an ongoing command [25].

## 5.3 Verification and validation

The process of verification and validation is relevant in all modeling projects. Once a model is implemented using a selected software tool, it must be debugged to ensure that it works correctly. In simulation literature, *model verification* is the process of determining that a model works as specified. [34] Eliminating bugs in a simulation model can be very time consuming, especially if it is implemented with a general purpose programming language, such as C++. In this work, a specialized simulation language (ProModel) was used. Because the structure of the language is limited, it reduces the amount of coding errors compared to a general purpose language. Verifying the model was done in two phases. In the first phase, the debugging capabilities of ProModel were used to find and eliminate the easily noticeable bugs. This was done continuously as the model was being built. ProModel provides a trace capability, which enables the user to

follow the events of the simulation to see if it is performing the way it should. Screen messages and animation were also used in this debugging process. Because the simulation model has a lot of decentralized logic, it is important to test various combinations of model input. This was done in the second phase of the model verification. Testing all the parameter combinations with their allowed ranges would have been too time-consuming and partly irrelevant for the purposes of this work. Therefore, most of the verification effort was put into the installed system type with different control parameters.

The process of determining the degree to which a model corresponds to the real system, or at least accurately represents its specification, is referred to as model *validation*. Proving absolute validity is a non-attainable goal. So actually validating a model is the process of substantiating that the model, within its domain of applicability, is sufficiently accurate for the intended application. [34] Validation is an inductive process through which the developer draws conclusions about the accuracy of the model based on the evidence available. Gathering evidence to determine model validity is largely accomplished by examining the model structure (i.e., the algorithms and relationships) to see how closely it corresponds to the actual system definition. [34]

The simulation tool includes some rather complex control logic, so the validation process was started with the graphic animation capabilities of ProModel. For example the crane's current location is updated on the screen as well as the number of totes it is carrying. The simulation can be paused at any time to check the values of global variables or the data in the arrays. The next step was to analyze the recorded output to observe if the results appear reasonable. The validation was done first for the simplest controls. For example FCFS sequencing was tested before the other algorithms so that they could be compared. The LP model was first verified separately after which it was validated with the simulation model. To help the validation, the LP model was programmed to log all cycle information in plain text, so that solution feasibility could be easily checked.

During this thesis the simulation model could not be fully validated against the installed system, due to a lack of operating data. During the making of the thesis, only a light unit test was performed for the system. This was a contractual test implemented according to an adaptation of the FEM 9.851 standard. A single preprogrammed test cycle was repeated five times by each of the cranes. This data was only helpful although not quite sufficient to validate mechanical parameters of the crane. It was the only data from which the crane handling time and crane positioning time could be obtained. The measured data was compared with the values written in the crane's technical specification. The average values for accelerations and load handling times could be calculated from the data quite accurately although there were some irregularities in the test cycle data, which could not be fully captured with the constant acceleration model. For example the vertical acceleration was found to be slightly higher when the crane was moving downwards, and the horizontal travel times were shorter when the crane was traveling away from the I/O point. These irregularities could not be explained with the loads the crane was carrying. Nevertheless, the calculated average horizontal and vertical accelerations were used

because they were significantly lower than the values in the technical specification. The maximum vertical and horizontal speeds could not be reliably determined from the data, because too few of the crane's movements were long enough to reach maximum speed. Therefore, the maximum speeds from the technical specification were used instead. The obtained values for the crane parameters were fed as an input to the simulation model where the exact same test cycle was then repeated. Although there were some differences in travel times, the total difference between the simulated and average measured cycle was less than 1 second. This difference was much less than the values obtained by using only the values in the specification. Simulating the test cycle also helped prove that the KPIs (key performance indicator), such as crane response time and cycle time, are programmed correctly in the model.

Due to a delay with a related project, production-like operation of the AS/RS could not be tested during the making of this thesis. Production testing would've provided important data for validating the controls of the system which are concerned with sequencing and cycle formation. As a consequence of the lack of this data, the supplier's algorithm could only be compared with a technical specification document where it is described. Although results obtained in this thesis cannot fully be linked with the installed system, the most important features are captured in the model which is enough to carry out a comparison of the control methods. As described in earlier chapters the crane control always involves a number of rule-based decisions for special situations. Not all of these were modeled due to lack of information from the supplier. These unmodeled rules include open location selection under a very high rack fill level (near 90 %), and handling retrieval requests with different priorities. The simulation scenarios are chosen in a way that these unmodeled rules wouldn't have an effect on the crane's operation.

Another validation method for the simulation results is to compare them with literature. This could be done only partially, because studies of the exact same system type weren't found. Even so, it was possible to utilize some of the results of other AS/RS studies to help with the validation process.

# 6 Experimental design and simulation results

This chapter will first go through the setup of the simulation runs, including choice of parameter values, followed by the simulation results. The test runs were designed in a manner which mimics the load variations of distribution center production. Two main workload scenarios can be found in almost every distribution center. Here they are called *beginning of shift* and *on shift*. They are used to measure the effect of two complementary control decisions, request sequencing and dwell point positioning, respectively.

## 6.1 Beginning of shift

**Description**

The beginning of shift scenario corresponds to a production situation, where a large amount of orders have arrived during off-shift and are waiting to be picked at the start of the next shift. The work schedule of a distribution center is usually paced according to the delivery route schedule. If there is a break in the delivery schedule, e.g. on Sunday, there is no use picking orders since they can't be dispatched. During off-shift, the orders are buffered, and when the next shift begins, they are released to picking either all at once, or in waves.

The main target of this scenario is to measure the efficiency of alternative sequencing and cycle forming strategies. To get the most accurate result, the scenario was defined so that both retrieval and storing queues were initially populated so that sequencing algorithms could start working right away without a warmup period. The initial number of retrieval requests in queue was set to 500, and the initial storing task queue was 25. The same set of randomly generated retrieval tasks was used for each run. No additional retrieval tasks or external storing tasks were introduced during the runs. In real production the storing task queue would initially be empty, unless a storage replenishment process were started immediately at the beginning of the shift. As orders are picked, the return flow of the storage totes generates storing tasks. This causes a continuous tote flow in both directions until the last retrieval task is executed. The 25 initial storing tasks are enough to start the continuous work of the crane and prevent the storing task queue from being exhausted during the simulation run. The crane works continuously at full capacity, until all of the retrieval tasks are completed. When the crane is working in this state of practically 100 % utilization, the significance of request sequencing is at its highest.

The most important performance indicator of this scenario is system throughput, which can here be measured from the total time that it takes to perform all the storing and retrieval tasks in queue. The total crane movement time will also be considered separately, because the handling time is such a large part of the cycle time. The time that a retrieval task spends in the infinitely large retrieval queue is not considered, since the retrieval requests are considered to have an equal priority. The time that a retrieval request spends in the retrieval sequencing queue is recorded so that possible differences between dynamic and static sequencing can be noticed.

**Rack fill level**

The rack fill level is an important parameter to study in the beginning of shift scenario, because it affects the possibilities to choose locations for storing and rearranging totes. In the installed system, all the compartmented totes will be stored in the racks even when they are empty, because they can't be stacked or used as order totes like the single-compartmented totes. The empty compartmented totes also add to the rack fill level, because they occupy locations just as filled totes. Additionally, some order totes can also be temporarily buffered in the rack if they are waiting for currently unavailable stock, but these should be quite scarce. The total portion of compartmented totes in the system is 21.3 %. In the beginning of shift scenario, two rack fill levels are considered: normal, and high. The normal production fill level is chosen to be 70 %. This is close to a desired fill level in production, when the compartmented totes are also considered [20]. The high fill level is set to 85 %. This leaves some margin to the supplier's upper bound of 90 %, but is considerably higher than the normal fill level. The initial situation of the storage racks was generated before simulation with the rack fill subroutine. To make the runs comparable, the exact same starting situation was used for all the runs with the same fill level. A distribution of the occupied and free storage locations for the two fill levels is presented in Figure 14.

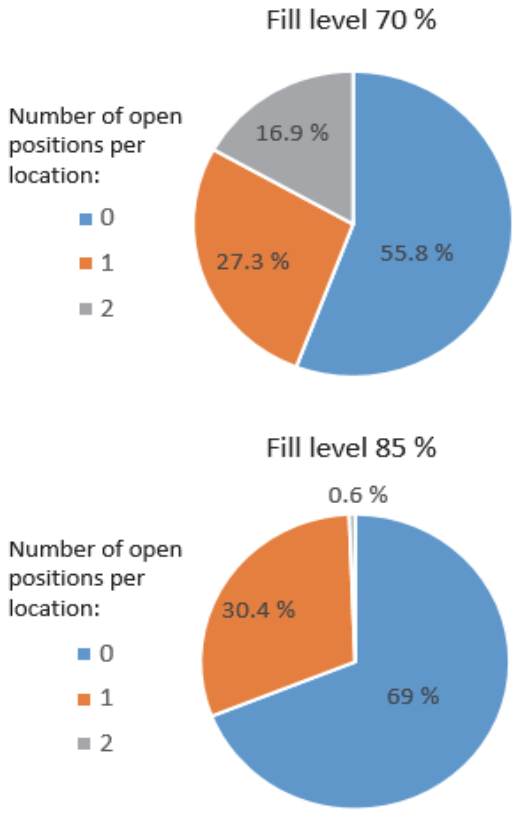


Figure 14: Rack location distribution, fill level 70 % (upper chart) and fill level 85 % (lower chart)

**Frozen horizon**

As was previously mentioned, the length of the sequencing horizon was fixed to 10, due to the physical restrictions of the system. This means that the frozen horizon $f$ can be an even number between 2 and 10. Three values of $f$ are considered in the runs: 2, 6 and 10. $f = 10$ is the static sequencing approach which should work well with the LP model. The value $f = 2$ is not considered an alternative for the LP model, because it defeats the purpose of optimizing over multiple cycles, and causes an excessive amount of calculation which would slow down the simulation. The TT heuristic does not optimize the cycles jointly as the LP model, but as a greedy heuristic it can still perform well with $f > 2$. Therefore, all three values of $f$ will be simulated with the TT heuristic. The supplier's algorithm does not try to optimize over multiple cycles, so only $f = 2$ needs to be considered for it.

## 6.2 On shift

The on shift production scenario means that the system has handled the task queues from the beginning of shift scenario and serves new tasks as they arrive. New tasks can be sent in waves or one by one. In the installed system, orders are waved at a higher level by the WMS (warehouse management system) are sent to the automatic storage system which is controlled by the WCS. The requests are then split to different picking areas including the five aisles of the AS/RS. Thus the waves from the WMS are broken down before they reach an individual crane. In this scenario the tasks arrive to the crane one by one. The arrivals of retrieval tasks and external storage tasks are independent. They are modeled as a Poisson process, whose intensity is the expected number of tasks per hour.

In the on shift scenario the importance of sequencing and dwell point positioning depend on the rate of the arriving requests. If the requests were buffered in the WCS and sent to the crane in larger waves, then sequencing might be more important. If no queues form during the shift, then sequencing will not have an effect on the operation.

The normal workload for the crane was calculated from the estimated production level for the system dimensioning year, which is two years after the completion of the installation. The tote flows in Table 4 are calculated for a single aisle. The average picking time was obtained from simulation results of the whole automatic warehouse system. The time to travel the infeed conveyor is reduced from the picking time, because the tote becomes an active storing task as soon as it arrives at the end of the infeed conveyor. Order totes are sometimes buffered in the miniload, because they are waiting for goods from another area. The leaving order totes are included in the tote empty probability. In this scenario the storage tote flow between aisles is assumed to be balanced. Only the 70 % rack fill level scenario will be simulated for the on shift scenario. The sequencing rule for this scenario will be the one used by the supplier. The sequencing rule and rack fill level are not expected to have much of an impact in this scenario, because the tote flows in Table 4 are considerably lower than the maximum throughput of the system. Thus, the

crane will not have a very high utilization rate. The simulation time for the scenario is set to 7 hours and there are no breaks during the shift.

Table 4: Tote flow parameters used in the on shift scenario

| parameter | value |
|---|---|
| retrieval requests per hour | 43.3 |
| external infeed: storing requests per hour | 5.25 |
| tote empty probability | 7.2 % |
| average picking time | 107 sec |
| wave size | 1 |

With the values from Table 4, the optimal dwell point positions were calculated according to formulas (7) – (9). The probabilities of a storing task, $p_s$ were calculated with equation (10). Because the optimal dwell point calculation formulas has not been extended to situations where the I/O point is not at the corner of the rack, the row height of the I/O point, which is 4 in the installed system, was simply added to the optimal dwell point row. This might cause a small deviation to the theoretical optimal dwell point. The calculated dwell point values are presented in Table 5. As can be noticed, the optimal dwell point shifts closer to the I/O point when the number of tote in picking increases. Already with 2 totes in picking $p_s > 0.5$, which makes the I/O point the optimal dwell point. Thus, the values for over 2 totes in picking do not need to be calculated.

Table 5: Optimal dwell points of the crane

| number of totes in picking | probability of a storing task, $p_s$ | optimal dwell point row | optimal dwell point column |
|---|---|---|---|
| 0 | 0.12 | 15 | 25 |
| 1 | 0.47 | 6 | 5 |
| 2 | 0.61 | 4 | 0 |

## 6.3 Simulation results

The results for the beginning of shift scenario are presented in tables 6 and 7. It is easy to see from the results that the FCFS rule, which is basically a no control scheme, performs worse than all of the algorithms with both fill levels. This result is expected, and it also supports the hypothesis that the three sequencing algorithms presented in Chapter 4 work as intended. The dispersion in average cycle times and total travel times is considerably higher in the 70 % fill level scenario. Another noteworthy result is the performance change of the algorithms in the different scenarios. The supplier's algorithm is the least

affected by the increase in the fill level. The LP and TT algorithms had a more notable performance drop with the higher fill level.

The 70 % fill level runs show that the sequencing approach used by the supplier results in a very high crane travel time. This is expected, since the only travel time that it seeks to minimize is the one between the second storing location and the first retrieval location. On the other hand the algorithm has a smaller total handling time than the other algorithms. This is because it utilizes the locations with two open positions very efficiently for storing and rearranging. With 70 % rack fill level, there were plenty of locations available for combining two tasks.

The TT heuristic seems to perform best with the frozen horizon length 6. An interesting notice is that the smallest frozen horizon $f = 2$ performs the worst with both rack fill levels. Even the static approach with $f = 10$ results in smaller travel times. This gives indication that the average cost of the cycle with the lowest cost becomes high, even though the rack locations and sets of storing and retrieval tasks are updated after each cycle. Overall the TT heuristic slightly outperforms the supplier's algorithm. With $f = 6$, the TT heuristic gives 2.1 % lower average cycle time than the supplier's approach.

The LP model gives the overall best results in average cycle time and total task completion time, with both rack fill levels. Even though $f = 6$ surprisingly gave a slightly better result in the 70 % fill scenario, the static approach with $f = 10$ is by far more consistent, and shall therefore be used for comparison with the other algorithms. In the normal fill scenario, the LP model resulted in a 5.3 % lower average cycle time and 4.4 % lower total time than the supplier's algorithm. The TT algorithm with $f = 6$ was second best, with a 2.1 % improvement in average cycle time compared to the supplier's algorithm. The average cycle time improvement of LP compared to the TT heuristic with $f = 6$ was 3.1 %.

Table 6: Simulation results of the beginning of shift scenario with fill level 70 %.

| sequencing algorithm | frozen horizon | average cycle time (s) | total crane travel time (s) | total time to complete tasks (s) |
|---|---|---|---|---|
| FCFS | - | 39.02 | 5854 | 11070 |
| supplier | 2 | 36.88 | 5967 | 10500 |
| TT | 2 | 36.48 | 5290 | 10358 |
| TT | 6 | 36.12 | 5203 | 10313 |
| TT | 10 | 36.22 | 5243 | 10335 |
| LP | 6 | 34.94 | 5304 | 10016 |
| LP | 10 | 35.01 | 5044 | 10035 |

In the 85 % fill scenario, the order of the algorithms remains the same, but the differences are much smaller. The average cycle time improvement of LP compared to the supplier's algorithm was 1 %, and for TT with $f = 6$, the improvement was 0.7 %.

Table 7: Simulation results of the beginning of shift scenario with fill level 85 %.

| sequencing algorithm | frozen horizon | average cycle time (s) | total crane travel time (s) | total time to complete tasks (s) |
|---|---|---|---|---|
| FCFS | - | 41.31 | 6002 | 11579 |
| supplier | 2 | 36.91 | 5781 | 10508 |
| TT | 2 | 37.13 | 5658 | 10519 |
| TT | 6 | 36.65 | 5558 | 10427 |
| TT | 10 | 36.81 | 5546 | 10486 |
| LP | 6 | 37.69 | 5693 | 10705 |
| LP | 10 | 36.57 | 5426 | 10410 |

Another interesting KPI in the beginning of shift scenario is the time that retrieval tasks spend in the sequencing queue. The recorded values for average and maximum stay times in the sequencing queue are presented in Table 8. As can be noticed, only the supplier's algorithm and the LP model with frozen horizon 10 can guarantee that all tasks pass the sequencing queue in a reasonably short time. Because an urgency rule was not incorporated in the TT algorithm, some retrieval tasks in the back of the rack can get profoundly delayed in the sequencing queue.

Table 8: Stay times of retrieval tasks in the sequencing queue.

| sequencing algorithm | frozen horizon | average stay time (s) | max stay time (s) |
|---|---|---|---|
| supplier | 2 | 207 | 426 |
| TT | 2 | 205 | 6475 |
| TT | 6 | 204 | 1884 |
| TT | 10 | 204 | 404 |
| LP | 6 | 198 | 879 |
| LP | 10 | 199 | 396 |

Figure 15 shows a time series of the tasks performed during the on shift scenario with the I/O dwell point strategy. The number of dwell point calls is also plotted in yellow. The total amount of dwell point calls during the 7 hour shift was 271. This accounted for 40.9 % of all the requests during the simulation. The crane utilization rate, which includes all operating time and travel to the dwell point, was highest (35.7%) with the I/O point strategy. According to previous studies, this should be a low enough utilization to determine the potential effect and differences between dwell point policies [25].

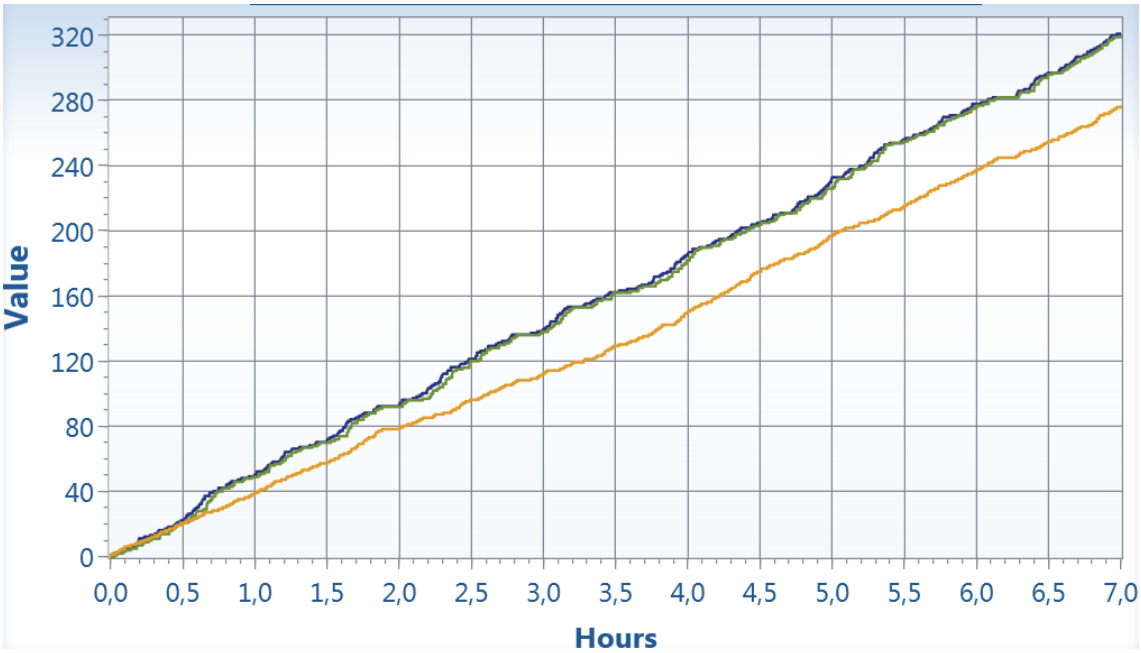


Figure 15: Number of totes stored (green), number of totes retrieved (blue), and number of dwell point calls (yellow) during the on-shift scenario with the I/O dwell point strategy.

The results of the on-shift scenario are presented in Table 9. The optimal dwell point strategy caused 8 % extra crane travel compared to staying at the last location. For the I/O strategy the addition was 9 %. The task arrival rates were so low that the length of both task queues peaked at 3 for all of the methods. This means that there was no chance to apply sequencing during this operational period, so the crane operated in FCFS mode.

Table 9: Comparison of dwell point rules in the on-shift scenario.

| | crane response time (s) | | request turnaround time (s) | | max queue length | total distance traveled (m) |
|---|---|---|---|---|---|---|
| | average | max | average | max | | |
| stay at last location | 4.87 | 11.08 | 20.11 | 91.41 | 3 | 9508 |
| return to I/O point | 5.33 | 11.07 | 20.24 | 93.25 | 3 | 10353 |
| optimal dwell point | 4.42 | 10.79 | 20.11 | 87.94 | 3 | 10303 |

The optimal strategy resulted in the lowest average crane response time. Compared to the optimal strategy, staying at the last location resulted in a 10 % higher average crane response time. The I/O point strategy gave the worst result, with a 20.5 % difference to the optimal strategy. For average request turnaround time, the maximum difference between all the strategies was under 1 %. The maximum request turnaround time was also lowest with the optimal strategy. The difference was 3.9 % to the stay-at-last-location strategy and 6 % to the I/O point strategy.

## 6.4 Analysis of control decisions

**Main findings**

The sequencing and cycle formation problem proved to be worth studying for the given system type. Although handling time accounts for roughly half of the cycle time, the impact of sequencing and cycle formation was found significant when there are many tasks in queue. The simple FCFS rule took 10 % more time to handle the same tasks as the LP model, which was found the most efficient strategy. The difference between the three algorithms presented in Chapter 4 was considerably larger when the rack fill level was 70%. A lower fill level gives the algorithms more alternatives for open location selection and task combination, thus increasing possibilities to save travel time. Another interesting notice is that shortening the frozen horizon did not necessarily lead to better results with the greedy TT heuristic. The TT algorithm gave best results with $f = 6$. This result gives reason to test the other two options ($f = 4$ and $f = 8$ ) as well.

One of the important questions going into the simulation experiments was, whether the benefit of optimizing over $\frac{h}{2}$ cycles with the static LP model would be enough to compete with the dynamic heuristic algorithms and TT in particular, since it has a similar cost function. According to the simulation results, the LP algorithm proves the optimization to be advantageous. This can be interpreted so that utilizing the combined information of the requests in the sequencing horizon can be more significant than the additional information obtained by updating the rack and sequencing queues after performing a part of the sequenced cycles.

The effect of the search neighborhood size of the LP algorithm could have been more thoroughly studied. A few experiments with search neighborhood size 100 were conducted and found to give practically the same results as 150. A more important factor seemed to be the choice of which open locations to include in the search neighborhood. Random selection proved to be pretty good, but there might be better options.

The on-shift scenario gave strong evidence that the dwell point strategy used in the installed system works well in comparison to the tested alternatives and also leads to the smallest total distance traveled by the crane. Additionally, the simulation results indicate that dwell point positioning only has a minor effect on the average request turnaround time with the estimated normal workload. Even though the average crane response time could be enhanced, the effect was not carried over to turnaround times. An explanation for this is that 59 % of the tasks arrived when the crane was active. Another factor which raised the average request turnover time is the relatively high (28 %) proportion of blocked retrieval tasks. Shortening the crane response time of a blocked retrieval task does not reduce its turnover time significantly, because it may still take three other crane movements to complete the task. Although the exact task rates were known in this case, the optimal dwell point strategy didn't dominate the other strategies. Only the I/O strategy was slightly weaker than the other two, which may be due to the long rectangular shape of the rack ($b = 0.47$). The evenness of the three strategies is supported by an earlier

study where random storage was found to be an equalizing factor [27]. One aspect which was bypassed in the found dwell point literature is the effect of the physical implementation, including the dimensions of the rack and the accelerations and speeds of the crane. If the time to travel the dimensions of the storage rack were very large, potentially more time could be saved with dwell point positioning. The earlier dwell point studies, in which larger effects were observed, mostly referred to slow moving cranes that handle heavy pallets. With fast miniload cranes like the ones in the installed system, at least the absolute time savings of dwell-point positioning are understandably smaller. Varying the tote-flow rates would of course affect the results. The highest effect of dwell-point selection, as well as the largest differences between dwell point policies, would be achieved with a very low system utilization. On the other hand, lowering system response times is generally more important when the system has more workload.

The on-shift scenario also indicates, that request sequencing won't have an effect if there aren't any queues and the tasks arrive individually with the simulated rates. In real production, the arrival pattern would probably not be as smooth even though the order waves from the WMS are broken down. There would probably be some situations where the task queues momentarily grow, giving sequencing a possibility to effect the crane movements.

**Enhancement ideas**

There are many possibilities to develop and improve the sequencing and cycle formation algorithms presented in this work. Some targets for development and ideas are presented in the following.

Open location selection proved to be a vital part of the sequencing and cycle formation algorithms. An indication of this is the decrease in performance when the fill level was increased. Especially the utilization of locations with two open positions to combine two storing or rearranging tasks seems to be important. The supplier's algorithm utilized these possibilities best, which made it perform reasonably well in comparison to the TT and LP algorithms. The LP model and TT heuristic lacked the capability to combine two rearrangement tasks in the same double-deep location. A modification for doing a double-rearrangement could be added to the cost functions of the two algorithms similarly as it was done with storing tasks. The storing location for the double-rearrangement could be chosen heuristically, e.g. the closest one to the second retrieval. This might slightly improve the performance of LP and TT, but overall the effect would be rather small, at least under a normal rack fill level, where only about 7 % of the cycles included two rearrangements. Based on the results with the 70 % fill scenario, it would also seem justified to always choose a storing location with two open locations over two separate locations. This modification could be easily made in both the TT and LP algorithms.

One aspect of the cycle formation problem, which was not fully captured by any of the methods, is routing with one or two rearrangement moves. As was noticed with performing a double storing move, the most potential routing improvements are the ones

that combine tasks so that two totes are deposited or picked up from one location. This could have been a source for more efficiency. For example, when there is one rearrangement task, the crane could combine the second storage task with the rearrangement task by depositing the two totes in the same location. This would save one crane movement and depositing time. Other routing options than store – store – (rearrange) – retrieve – (rearrange) – retrieve could be added to the TT heuristic as alternative distance metrics. However, the feasibility of the routing would have to be checked individually for each cycle. This would increase the time complexity of the heuristic, which is already quite large.

In real production the prioritization of tasks could be done in an upper system so there could be constraints that certain tasks need to be performed before others. These priority constraints would be simple to add to the TT heuristic as well as the supplier's algorithm because they form the cycles sequentially. The LP model could not handle different priority tasks because it doesn't control the order of the cycles.

If the choice of which storage tote to retrieve for each task could be made purely based on cycle time optimization, this decision would be worth including in the sequencing and cycle formation problem. All of the presented algorithms would have to be extended to make this decision. With a large LTPR, this choice could be important in reducing cycle times.

Finally, if the TT algorithm were considered for production use, it should be made sure that tasks don't get excessively delayed in the sequencing queue. This could be achieved by adding a simple urgency rule which would function in the following way: After every sequencing calculation, the retrieval sequencing queue is checked for tasks which have been waiting over a pre-defined maximum time in the sequencing queue. If so, the oldest one would be chosen to be performed in next cycle.

# 7 Conclusions

The purpose of this thesis was to study alternative crane control methods of a double-deep dual-shuttle automatic storage and retrieval system. The study was focused on an AS/RS which is currently being implemented in an ongoing warehouse automation project. The goal was to find out, how efficiently the control policies implemented by the supplier work, and how they could be improved. Crane control can potentially improve the performance of any AS/RS without making mechanical changes to the system. A miniload AS/RS in a distribution center can operate every day round-the-clock with varying load, which makes even small improvements worth considering. All crane control policies essentially try to utilize available information contained in currently known storing and retrieval tasks, SKU features, and estimated tote flow rates, to decide where and how the crane should move. The control problems are strongly dependent on the system type, which means that the results of this thesis can only be connected to the studied configuration.

A discrete-event simulation tool was built so that the effect of various system and control parameters could be compared. The simulation model was programmed with ProModel. It was designed to enable study of multiple different types of AS/RS. In this work the tool was used to study the AS/RS related to the ongoing project. To test the effect of the control rules, the system was simulated under two different workload scenarios and rack fill levels.

Storage location assignment is a mandatory crane control decision, which has been shown to have a significant effect on expected retrieval times, at least with single-deep racks. Deciding on a storage policy should be done based on the availability and level of accuracy of SKU information. In this thesis, no information about future SKUs was available, so the assessment of the alternative storage policies was done based on literature and expert opinions. Random storage, which is used by the system supplier, is a good choice when the amount of SKUs is very large and the items in store change often. It also works well with double-deep racks, because rearrangements can be performed in any open location, which enables combining a rearrangement with a storing task or another rearrangement. If accurate and reliable information about the turnover rates for SKUs is available, then full-turnover storage or class-based storage can reduce average retrieval times compared to random storage. With a turnover-based item ranking, the effect of the policy strongly depends on the distribution of the storage tote flow in the system. The more skewed it is, the more potential improvement can be achieved compared to random storage. Dedicating a storage area to an item or a class of items requires extra rack space. Maintaining the integrity of a class-based or full-turnover storage policy is more difficult with double-deep racks because of forced rearrangements. Fast moving items should not block other fast movers or get blocked by slower moving items. Dedicating a zone of locations to an item or a class of items also removes much of the benefit of combining two storing or rearrangement tasks in a cycle, since the

probability that two sequential totes are allowed to be stored in the same location would be very small.

Three dwell point selection rules for the installed system were compared in this thesis. In addition to two static stay-at-last-location and returning to the I/O point rules, an optimal dwell point strategy based on expected tote flows was tested. The dwell point-strategies were compared with the simulation model in the on shift scenario, which mimics a steady production situation with no initial queues. According to the results, the optimal dwell point strategy could reduce the average crane response time by 10 % compared to the policy used by the supplier. However, the difference was practically non-existent when average request turnover time was used as a measure. The optimal strategy also resulted in 9 % more crane travel compared to the simple stay-at-last-location method used by the supplier. Because the optimal dwell point strategy is based on the assumption that the rates of tote flows are known, it would not be a viable option without sufficient information about production patterns.

The sequencing and cycle formation problem was broken down into sub-problems and three alternative sequencing algorithms were presented. Trying to optimize a combination of all the choices in each sub-problem proved to be too costly in computation time. The search space of the problem was reduced by limiting the amount of open locations and by making some of the choices heuristically before calculating the actual sequence. The LP algorithm developed in the thesis had quite many simplifications and restrictions, but treated the problem more comprehensively than the other algorithms. The LP algorithm was found computationally feasible in the context of this thesis. Furthermore, it outperformed both the TT heuristic and the algorithm used by the supplier. According to the simulation results of the beginning of shift scenario with a 70 % fill level, the LP algorithm was able to reduce the average cycle time by 5.3 % compared to the algorithm used by the supplier.

Although proven to be a computationally feasible option, implementing the LP algorithm in a real system would require a lot of customization and testing, before it could be used. Maintainability is an important criterion in control algorithms for warehouse systems [17]. Managing the program code of an optimization algorithm is significantly more difficult compared to the studied heuristic options. The LP algorithm is also very specifically designed for the studied configuration. Slight technical or configurational changes, such as adding an I/O point or having two independently operable shuttles on the crane, would change the problem, which would require a lot of reprogramming.

Overall, this thesis succeeded in fulfilling its objectives. Alternative crane control methods were studied, algorithms for control decisions were developed, and the implemented controls were successfully tested with the simulation model. The available information of the system was used for the parametrization and validation of the model. Although the validation could not be completed during this thesis, the simulation scenarios provided a good quantification of the control effects.

**Future development**

Validation of the simulation model can be continued when production-like testing is carried out with the installed AS/RS. First, more data is needed to validate the kinematics of the model. The speeds and accelerations need to be re-evaluated once there is a statistically significant amount of operating data. To test the cycle formation capabilities and maximum throughput of the system, a test similar to the beginning of shift scenario will be conducted. The infeed conveyor will be filled prior to the test and the crane will be given a set of equally urgent retrieval tasks to perform. This test can then be recreated in the simulation model with precisely the same initial storage rack state and set of retrieval tasks. This will help to validate the supplier's sequencing and cycle formation algorithm.

The simulation tool can be used in the future to analyze other AS/RS types in different projects with little or no modification. Modified versions of the control methods used in this thesis could prove even more beneficial for other system types. For example the study of pallet high-bay storage systems with unit-load AS/RSs could lead to very different findings compared to this thesis. An interesting extension to the simulation model would be adding SKU information. It could be read from a spreadsheet or database during the model initialization logic. This would open possibilities to accurately test class-based or full-turnover storage assignment policies.

# References


[1] K. J. Roodbergen and I. F. a. Vis, "A survey of literature on automated storage and retrieval systems," *Eur. J. Oper. Res.*, vol. 194, no. 2, pp. 343–362, 2009.

[2] S. Kulturel, N. Ozdemirel, C. Sepil, and Z. Bozkurt, "Experimental investigation of shared storage assignment policies in automated storage/retrieval systems," *IIE Trans. (Institute Ind. Eng.*, vol. 31, no. March 2015, pp. 739–749, 1999.

[3] R. Shell, *Handbook Of Industrial Automation*. CRC Press, 2000.

[4] R. Manzini, M. Gamberi, and A. Regattieri, "Design and control of an AS/RS," *Int. J. Adv. Manuf. Technol.*, vol. 28, pp. 766–774, 2006.

[5] J.-P. Gagliardi, J. Renaud, and A. Ruiz, "Models for automated storage and retrieval systems: a literature review," *Int. J. Prod. Res.*, vol. 50, no 24. February 2015, pp. 7110–7125, 2012.

[6] Y.A. Bozer and J.A. White, "Travel Time Models for Automated Storage/Retrieval Systems," *IIE Trans.*, vol. 16, no. 4, pp. 329–338, 1984.

[7] A. Keserla and B. Peters, "Analysis of dual-shuttle automated storage/retrieval systems," *J. Manuf. Syst.*, 1994.

[8] J. P. van den Berg and a. J. R. M. Gademann, "Simulation study of an automated storage/retrieval system," *Int. J. Prod. Res.*, vol. 38, no. 6, pp. 1339–1356, 2000.

[9] T. Atz, D. Lantschner, and W. A. Günthner, "Simulative throughput calculation for storage planning", Institute of Materials Handling, Material Flow, Logistics, Technische Universität München, 2013 [Online]. Available: http://www.fml.mw.tum.de/fml/images/Publikationen/2013%2008%2021_PAPER_SIMULATIVE%20THROUGHPUT%20CALCULATION%20FOR%20STORAGE%20PLANNING.pdf [Accessed: March 30, 2015].

[10] "Mini Load Automated Storage & Retrieval System" [Online]. Available: http://daifukuna.com/Products/Automated-Storage-Retrieval-System-AS-RS/Mini-Load-Automated-Storage-Retrieval-System. [Accessed: April 27, 2015].

[11] J. Pazour and R. Meller, "Modeling the Inventory Requirements and Throughput Performance of Picking Machine Order-Fulfillment Technology", *Progress in Material Handling Research: 2012,* 2012.

[12] T. Lerher, M. Sraml, I. Potrc, and T. Tollazzi, "Travel time models for double-deep automated storage and retrieval systems," *Int. J. Prod. Res.*, vol. 48, no. 11, pp. 3151–3172, 2010.

[13] R. D. Meller and A. Mungwattana, "Multi-shuttle automated storage / retrieval systems," no. 1222, 1997.

[14] H. F. Lee and S. Schäffer, “Retrieval sequencing for unit-load automated storage and retrieval systems with multiple openings,” *Int. J. Prod. Res.*, vol. 34, no. 10, pp. 2943–2962, 1996.

[15] J. Gagliardi, J. Renaud, and A. Ruiz, “On sequencing policies for unit-load automated storage and retrieval systems,” *Int. J. Prod Res.*, vol. 52, pp. 1090-1099, 2014.

[16] H. F. Lee, “Performance analysis for automated storage and retrieval systems,” *IIE Trans.*, vol. 29, no. January 2015, pp. 15–28, 1997.

[17] F. Wahlström, Managing Director, Softsys Oy. Interview. Helsinki 21.8.2015.

[18] W. H. Hausman, L. B. Schwarz, and S. C. Graves, “Optimal Storage Assignment in Automatic Warehousing Systems,” *Manage. Sci.*, vol. 22, no. 6, pp. 629–638, 1976.

[19] J.-P. Gagliardi, J. Renaud, and A. Ruiz, “A simulation modeling framework for multiple-aisle automated storage and retrieval systems,” *J. Intell. Manuf.*, 2012.

[20] P. Korpiharju, Managing Director, EP-Logistics Ltd. Interview. Helsinki, Finland, 10.8.2015.

[21] C. J. Malmborg, “Storage assignment policy tradeoffs,” *Int. J. Prod. Res.*, vol. 34, no. 2, pp. 363–378, 1996.

[22] M. Kofler, A. Beham, S. Wagner, M. Affenzeller, and W. Achleitner, “Re-warehousing vs. healing: Strategies for warehouse storage location assignment,” *LINDI 2011 - 3rd IEEE Int. Symp. Logist. Ind. Informatics, Proc.*, pp. 77–82, 2011.

[23] M. J. Rosenblatt and A. Eynan, “Deriving the Optimal Boundaries for Class-Based Automatic Storage/Retrieval Systems,” vol. 35, no. 12, pp. 1519–1524, 1989.

[24] J.-P. Gagliardi, J. Renaud, and A. Ruiz, “On storage assignment policies for unit-load automated storage and retrieval systems,” *Int. J. Prod. Res.*, vol. 50, no. 3, pp. 879–892, 2012.

[25] R. D. Meller and a. Mungwattana, “AS/RS dwell-point strategy selection at high system utilization: A simulation study to investigate the magnitude of the benefit,” *Int. J. Prod. Res.*, vol. 43, no. February 2015, pp. 5217–5227, 2005.

[26] B. C. Park, “An optimal dwell point policy for automated storage/retrieval systems with uniformly distributed, rectangular racks,” *Int. J. Prod. Res.*, vol. 39, no. August 2000, pp. 1469–1480, 2001.

[27] P. J. Egbelu and C. T. Wu, “A comparison of dwell point rules in an automated storage/retrieval system,” *International Journal of Production Research*, vol. 31. pp. 2515–2530, 1993.

[28] B. A. Peters, J. S. Smith, and T. S. Hale, “Closed form models for determining the optimal dwell point location in automated storage and retrieval systems,” *Int. J. Prod. Res.*, vol. 34, no. 0020, pp. 1757–1771, 1996.

[29] H. F. Lee and S. Schäffer, “Sequencing Methods for Automated Storage and Retrieval Systems with Dedicated Storage,” *Iternational J. Ind. Eng. Comput.*, vol. 32, pp. 351–362, 1997.

[30] X. Tran, T. Tran, and H. Kim, “Local Search for Sequencing of Storage and Retrieval Requests in Multi-Shuttle Automated Storage and Retrieval Systems,” vol. II, 2014.

[31] B. R. Sarker and P. S. Babu, “Travel time models in automated storage/retrieval systems: A critical review,” *Int. J. Prod. Econ.*, vol. 40, pp. 173–184, 1995.

[32] K. Wu, S. S. Xu, and T. Wu, “Optimal Scheduling for Retrieval Jobs in Double-Deep AS / RS by Evolutionary Algorithms,” vol. 2013.

[33] R. Bosch and M.Trick, “Integer Programming,” in *Handbooks in Operation Research and Management Science*, 1989, pp. 69–95.

[34] M. Cashbook, M. Express, and M. Gold, “ProModel User Guide,” 2006.

[35] N. Van Omme, L. Perron and V. Furnon, "Google OR-Tools User’s Manual". Google, 2014.

[36] “COIN-OR” [Online]. Available: http://www.coin-or.org/. [Accessed: July 25, 2015].

[37] J. E. Mitchell, “Branch-and-Cut Algorithms for Combinatorial Optimization Problems,” *Handb. Appl. Optim.*, pp. 65–77, 2002.